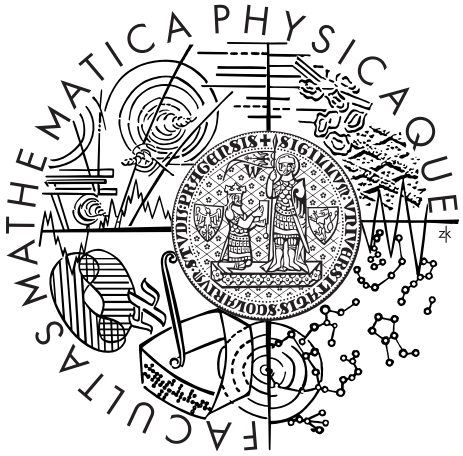
Univerzita Karlova v Praze
Matematicko-fyzikální fakulta

DIPLOMOVÁ PRÁCE



Jakub Michalko

Algoritmy detekce obchodních dokumentů podle šablon

Katedra softwarového inženýrství

Vedoucí diplomové práce:	doc. RNDr. Tomáš Skopal, Ph.D.
Studijní program:	Informatika
Studijní obor:	Softwarové systémy

Praha 2016

Chcel by som poďakovať pánovi Petrovi Koláčkovi so spoločnosti COST CUTTING solutions, ktorý mi predviedol software ReadSoft INVOICES a poskytol podnetné informácie týkajúce sa problematiky spracovávania oskenovaných dokumentov, ich prevodom to textu a extrakciou dát. Ďalej by som sa chcel poďakovať svojmu vedúcemu diplomovej práce doc. RNDr. Tomášovi Skopalovi podnetné rady a pomoc pri tvorbe tejto príce. V neposledom rade chcem tiež poďakovať svojej manžeke Marii Michalko za kontrolu práce a jej podporu.

Prohlašuji, že jsem tuto diplomovou práci vypracoval(a) samostatně a výhradně s použitím citovaných pramenů, literatury a dalších odborných zdrojů.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona v platném znění, zejména skutečnost, že Univerzita Karlova v Praze má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle §60 odst. 1 autorského zákona.

V ..... dne ..... Podpis autora

Název práce: Algoritmy detekce obchodních dokumentů podle šablon

Autor: Jakub Michalko

Katedra: Katedra softwarového inženýrství

Vedoucí diplomové práce: Tomáš Skopal, Assoc. Prof.

Abstrakt: Diplomová práca sa zaoberá analýzou a návrhom systému pre automatické rozpoznávanie dokumentov. Systém spracuje dokument a prevedie ho do textovej podoby, pričom musí byť zachovaná informácia o pôvodnej polohe slova v dokumente. Tieto dáta budú následne preskúmané a určitým dátam bude pridelený ich význam. Spôsob, akým bude dátam pridelený význam bude založený na pravidlách, ktoré môže meniť užívateľ podľa svojej potreby. Následne na podľa dát, ich pridelenému významu a ich polohe, systém nájde podobný dokument a podľa neho identifikuje aktuálne skúmaný dokument.

Klíčová slova: pološtrukturované dokumenty, annotácia,OCR, vyhľadávanie dokumentov

Title: Algorithms for business document detection using templates

Author: Jakub Michalko

Department: Department of Software Engineering

Supervisor: Tomáš Skopal, Assoc. Prof.

Abstract: Thesis deals with analysis and design system for automatic document recognition. The system examines the document and converts it into text data, and shall be preserved information about the initial position of the word in the original document. These data will then be reviewed and some of them will be assigned their importance. The way the data will be assigned is based on rules which may vary according to user needs. Subsequently, according to the data of their assignment and the importance of their position, the system finds a similar document and, if it identifies the current document examined.

Keywords: semistrucutred documents, annotation, OCR, document search

Obsah

1. Úvod

Dokumenty sa v dnešnej dobe nachádzajú buď v papierovej, alebo elektronickej podobe. Nevýhodou papierových dokumentov je, že prístup k nim je veľmi neefektívny a v medzinárodnej organizácií skoro nemožný, nehovoriac o nákladoch s tým spojených. Prevod do elektronickej podoby tieto problémy odbúrava, avšak v súčasnej dobe časť nestačí len oskenovať dokument ako obrázok a uložiť ho niekam na zdielané úložisko. Väčšina užívateľov totiž nechce len hľadať informáciu v jednom dokumente, ale chce vyhľadávať informáciu nad niekoľkými dokumentami. To samozrejme nie je možné bez toho, ak by dokument nebol prevedený do textovej podoby, respektíve do do podoby, v ktorej je vyhľadávanie možne. Korporátna sféra zase vyžaduje prevod papierových dokumentov do softvérového systému, s ktorým pracuje.

Každá organizácia rieši zadávanie dokumentov do interného systému po svojom. Bežnou praxou je najať si pracovnú silu, ktorá prepisuje dokumenty do interného systému. Tento spôsob je najlepší v prípadoch, keď dokumenty sú písane ručne. Iné organizácie zase nechávajú túto činnosť na klientoch. Najčastejšie je to vyplnenie formulára umiestneného na webových stránkach. V prípade, kedy sa jedná o dokumenty generované systémom, alebo nie je chcené, aby klienti zadávali dokumenty do systému, sa ako najlepšie riešenie naskytuje spracovať dokument softvérovo a ten zaniesť do interného systému automaticky.

Softvérové spracovanie tlačených dokumentov je stále sa rozvíjajúce odvetvie. Základom pre spracovanie tlačeného dokumentu je OCR systém, ktorý v obrázku dokumentu identifikuje znaky a tie prevedie na odpovedajúce znaky v elektronickej podobe.

1.1 Motivácia

Na trhu existuje mnoho komerčných riešení (niektoré sú uvedené v kapitole 2.2), no stále nie sú dokonalé. Ich know-how si prísne strážia. Všetky sú navyše špecializované na konkrétny druh dokumentu, ako napríklad faktúra, alebo životopisy. Anotovaniu dokumentov a dolovaniu dát z dokumentov sa venuje celé odvetvie informatiky [1], no predovšetkým je zamerané na anotovanie neštrukturovaných dokumentov.

V tejto práci sa k analýze dokumentu a pristupuje ako k analýze ľubovolnému typu dokumentu. Prednostne ale budeme vychádzať z analýzy korporátnych dokumentov.

1.2 Cieľ práce

Hlavným cieľom tejto práce je analyzovať dokument. Na základe tejto analýzy následne vyhľadať čo najpodobnejší spracovaný dokument (dokument, ktorý je spracovaný možno brať ako šablónu) a podľa neho v dokumente určiť, aký význam majú analyzované časti textu.

Unikátny spôsob vyhľadávania podobnosti dokumentov tkvie vo vzájomnej polohe blokov dokumentu textu. Blokom textu sa rozumie skupina slov, alebo

[1] respektíve časť odvetvia *information retrieval* zaoberajúce sa dolovaním dát

riadkov, ktoré spolu nejakým spôsobom súvisia. Mnoho dokumentov má podobné rozloženie blokov textu v dokument (napríklad adresa zákazníka býva v pravej hornej časti dokumentu). Práve tento spôsob vyhľadávania umožňuje priradiť význam jednotlivým blokom textu aj dokumentu, ktorý by iné softvérové riešenia považovali za dokument tlačený z inej šablóny.

Anotácia textu znamená priradiť celému textu, alebo jeho častiam význam, ako napríklad meno, priezvisko alebo telefónne číslo. Cieľom práce bude návrh a implementácia algoritmov, ktoré sa budú snažiť anotovať časti textu od jednotlivých slov až po bloky textu. Spôsobu spracovania slov od najmenších častí po najväčšie sa nazýva *zdola nahor*. Na základe anotovaného textu sa bude aplikácia snažiť vyťažiť dôležité dáta dokumentu. Podľa typu anotovaných blokov textu, ich vzájomných polôh a podľa už analyzovaných dokumentov sa aplikácia bude snažiť určiť typ dokumentu.

1.3 Štruktúra práce

Táto práca pozostáva z niekoľkých kapitol:

- V kapitole analýze existujúcich riešení si popíšeme celosvetovo najrozšírenejšiu aplikáciu, ktorú porovnáme s aplikáciou z aplikáciu od slovenských autorov.
- V kapitole popisujúce naše riešenie si okrem predstavenia samotného princípu popíšeme aj problematiky, ktoré bolo nutné vyriešiť.
- V kapitolách užívateľskej dokumentácie sú návody na inštaláciu aplikácie a jej spustenie

2. Existujúce riešenia

Podstatou tejto práce je navrhnúť spôsob, ako dáta v papierovej podobe preniesť do elektronického systému. V tejto kapitole si popíšeme existujúce spôsoby a softwarové riešenia, ktoré to dokážu.

2.1 Ručné zadávanie

2.1.1 Prepis

Medzi najstarší a stále používaný spôsob zadávania dokumentov do systému je ručné zadávanie. V praxi pri zadávaní väčšieho počtu dokumentov do systému sa najme lacná pracovná sila, ktorej zaškolenie vyžaduje krátky čas. Tento spôsob sa uprednostňuje v situáciách, kedy je treba zadať do systému dokumenty buď malý počet dokumentov, kedy sa spoločnosti neoplatí kupovať si drahé riešenia, alebo v prípadoch, kedy softvér nedokáže správne spracovať dokumenty. Väčšinou sa jedná o ručne písané dokumenty, alebo ručne vyplnené dotazníky. Nevýhodou tohoto spôsobu sú veľké časové a finančné náklady pri prepise strojovo generovaných dokumentov, ktoré sú softvérovo spracovateľné.

2.1.2 Vyplňovanie elektronického formulára

Iným spôsobom, ako zaviesť dokument do systému je nechať samotného užívateľa (respektíve odosielateľa) vyplniť elektronický formulár, ktorý je jednoducho spracovateľný systémom. Takéto formuláre sa často predstavujú napríklad dotazníky, registračné formuláre uchádzača o prácu, alebo vyplnenie objednávky v internetovej predajni.

Výhodou tohoto spôsobu je okamžité uloženie dokumentu do systému.

Nevýhodou je rôznorodosť rozhraní a dokumentov. Každý správca systému si to robí takzvane "po svojom". Výsledkom toho je, že jeden typ dokumentu/objektu, napríklad karta kontaktu, má v Google contacts jednu štruktúru, v Microsoft exchange druhú a v Skype opäť inú štruktúru. V ideálnom prípade by existoval jeden štandard popisujúci všetky typy objektov/dokumentov a jedna aplikácia, pomocou ktorej by užívatelia mohli zadávať tieto objekty/dokumenty. Celonárodné a medzinárodné organizácie zaoberajúce sa štandardizovaním typov dokumentov, vytvárajú protokoly a smernice, ktoré umožňujú priamu komunikáciu medzi rôznorodými systémami. V praxi ale nie každý investuje svoje prostriedky do implementácie rozhrania splňujúce štandardy a to buď kôli ich neznalosti, alebo kvôli množstvu nákladov, ktoré by musel na ne vynaložiť.

2.2 Automatické rozpoznávanie

2.2.1 Datamolino

Datamolino je online služba, ktorej vstupom sú oskenované dokumenty v podobe pdf súborov, alebo obrázkov a výstupom sú extrahované dáta, ktoré je možné buď importovať do účtového systému, alebo stiahnuť v podobe CSV alebo XSL

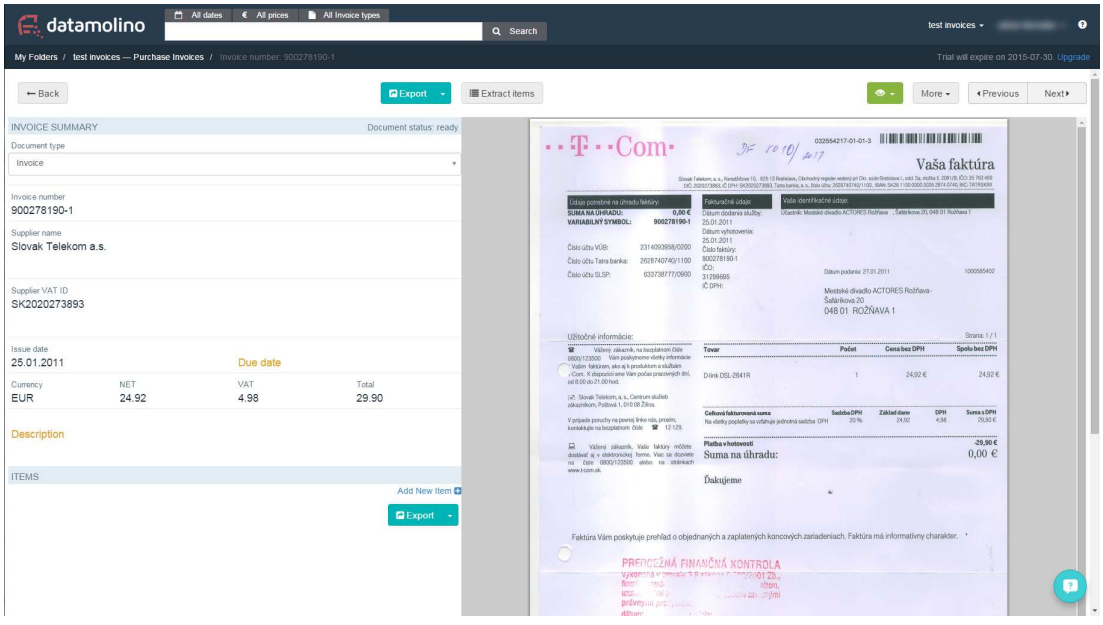
súboru. Túto službu je možne si bezplatne vyskúšať po dobu 7 dní. Špecializujú sa na faktúry, profomy faktúr, dobropisy a účtenky. Ich zámerom je oprostiť užívateľa od akejkoľvek nutnosti zásahu do procesu rozoznávania. To vedie k nutnosti čakať 1 až 3 dni na spracovanie. Tento startup projekt sa snaží konkurovať existujúcim službám cenou a odlišným prístupom k spracovaniu dát a možnosťou exportovať dáta do mnohých účtovných systémov.

Po nahraní dokumentu na vzdialený server sa automaticky spustí rozoznávanie. Systém sa snaží automaticky detekovať, o aký typ dokumentu sa jedná z neho vyextrahovať dôležité dáta. Rozoznané dokumenty môžu skončiť v troch stavoch:

- *rozpoznané*
- *1 deň*
- *3 dni*

Na testovacích dátach dokázal systém automaticky [1] vyextrahovať číslo faktúry, IČO odosielateľa a celkovú čiastku faktúry. V ostatných prípadoch bolo nutné počkať, než operátori potvrdia/doplnia údaje na faktúre. Systém automaticky nerozoznáva položky na faktúre. Pri testovaní v niektorých prípadoch namiesto položiek faktúry vyplnil *popis faktúry*, do ktorého vložil text popisujúci prvú položku na faktúre. V jednom prípade dokonca s preklepom a nesprávnym formátom dátumu (na niektorých miestach chýbala medzera). Pre doplnenie položiek musí užívateľ kliknúť na tlačítko rozoznať položky. Tým sa opäť faktúra dostala do stavu *1 deň* alebo *3 dni*. Extrahované položky mali správne vyplnené meno položky, ale chýbali údaje o cene a DPH. Ďalšou nevýhodou je, že nezistíte, z ktorého miesta v obrázku dáta pochádzajú.

Po technickej stránke aplikácia nie je na špičkovej úrovni, ale pre potreby zákazníkov postačuje.



Obrázek 2.1: Datamolino

[1] Dokument neskončil v stave 1 deň alebo 3 dni.

2.2.2 LANA

LANA aplikácia, ktorá bola vyvinutá na Matematicko-Fyzikálnej fakulte Karlovej Univerzite ako softvérový projekt, kde som bol členom vývojového tímu. Aplikácia oskenovaný dokument najprv skonvertovala do obrázku [2], ktorý následne vyrovnala a pomocou aplikácie Tesseract extrahovala text s jeho umiestnením v dokumente. Slová z extrahovaného textu postupne spájala do riadkov a riadky do blokov textu podľa toho, aká veľká medzera ich oddeľovala.

Hlavnou myšlienkou celého algoritmu rozoznávania bolo, porovnávanie aktuálneho dokumentu s dokumentami, ktoré boli správne spracované a v nich boli identifikované dôležité dáta ako číslo faktúry, adresa odosielateľa atp. Zhoda dvoch dokumentov záležala na tom, ako sa prekrývajú jednotlivé bloky textov. K tomu bolo potrebné normovať všetky dokumenty do jednotnej šírky, pretože veľkosť prekrytia blokov bola počítaná na základe ich súradníc. Pretože ak by bol jeden dokumenty naskenovaný v rôznych rozlíšeniach, tie isté bloky by mali iné súradnice a prekrývali by sa len čiastočne, alebo vôbec.

Aplikácia podľa nájdeného podobného dokumentu dokázala následne určiť, o aký typ dokumentu sa jedná a identifikovať miesta, kde sa dôležité dáta nachádzajú. V porovnaní s Datamolino je LANA na rovnakej úrovni. Lana umožňuje samotným klientom doopraviť dáta, zatiaľčo Datamolino má na to interných operátorov.

2.2.3 ReadSoft INVOICES

Spoločnosť ReadSoft sa vo všeobecnosti zaoberá všetkým, čo súvisí so spracovaním dokumentov a ich workflow. Jedným z ich produktov je služba ReadSoftInvoices, ktorá spracováva naskenované dokumenty.

Umožňuje prepojenie s inými účtovnými systémami. Dokáže spracovávať akýkoľvek typ dokumentu. Pre čo najlepšie extrahovanie textu z dokumentu používa až 3 ICR enginy[3]. V prípade nejednoznačnosti rozoznaného znaku, porovnáva výsledok s ďalšími dvoma. ICR umožňuje rozoznať nie len tlačené, ale aj ručne písané znaky.

Nové (nenaučené) typy dokumentov je možné ho naučiť, čím sa vytvorí takzvaná šablóna dokumentu. Podľa týchto šablón následne systém dokáže určiť miesta, kde sú ktoré informácie umiestnené.

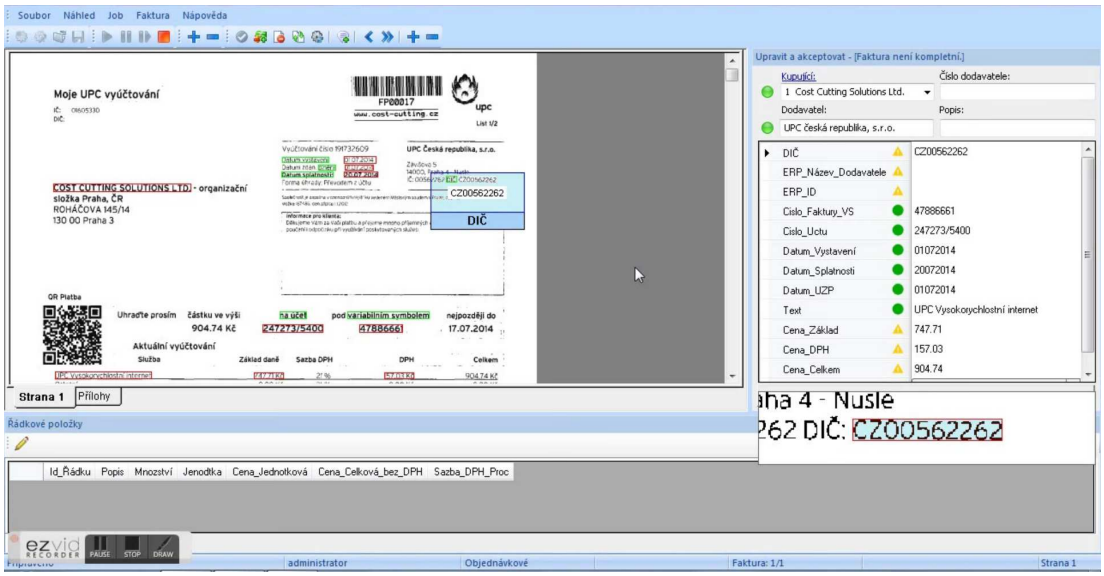
Okrem toho sa v nenaučených dokumentov snaží systém nájsť dôležité informácie podľa kľúčových výrazov, ako napríklad *DIČ*, alebo *Daňové identifikační číslo*. V prípade, že dané kľúčové slovo, snaží sa nájsť v jeho bezprostrednom okolí (Väčšinou napravo od, alebo pod kľúčovým slovom), jeho hodnotu, ktorá odpovedá významu kľúčového slova. Príkladom je DIČ českej spoločnosti začína znakmi CZ a pokračuje číslami (alebo aj znakmi), rodnému číslu odpovedá 6 čísel, znak / a 4 čísla atp. .

Aplikácia je platené a nie je možné si ju bezplatne vyskúšať. Pri prezentácií aplikácie bolo povedané, že pre čo najlepšie výsledky je tiež potrebné kvalitné sekenre. Aplikácia bola nastavená na český jazyk. Keďže naše testovacie faktúry boli v slovenčine a v trochu horšej kvalite [4], bolo nám povedané, že na takýchto

[2] V prípade, že vstupným dokumentom bol pdf dokument.

[3] ICR je zkratka *Intelligent character recognition*, čo je pokrokovejšia verzia OCR

[4] testovacie dáta sú súčasťoou priloženého CD



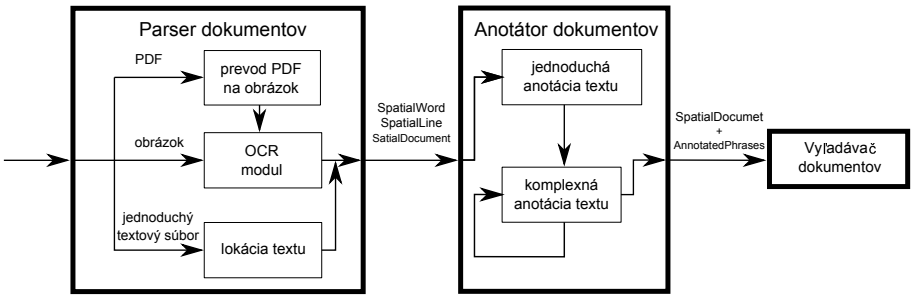
Obrázek 2.2: Readsoft

dokumentoch to nebude pracovať správne a teda nevieme, aké výsledky by sme dostali nad týmito dátami. Ale podľa popisu a prezentačných videí, je ReadSoft INVOICES technologicky najvyspelejšou z pomedzi analyzovaných aplikácií v tejto práci.

Porovnanie obecnejších vlastností uvedených aplikácií je možné nájsť v tabuľke 5.1.

3. Model a implementácia riešenia

3.1 Model

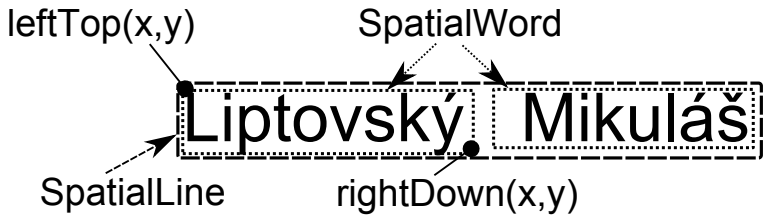


Obrázek 3.1: Model

Parser dokumentov zo vstupného dokumentu extrahuje text a jeho lokáciu. Výsledné dáta sa označujú ako **SpatialDocument**. **Anotátor dokumentov** v tomto dokumente vyhľadá podľa definovaných fráz a umiestnení slová, ktorým prídeli ich význam (anotáciu). Výsledok je porovnávaný vo **Vyhľadávači dokumentov**, ktorý vyhľadá podobný dokument a pravdepodobnosť, na koľko sú tieto dokumenty podobné. Z nájdeného podobného dokumentu potom anotátor určí, o aký typ dokumentu sa jedná a ktoré anotované fráze sú dôležité.

3.2 Parser dokumentov

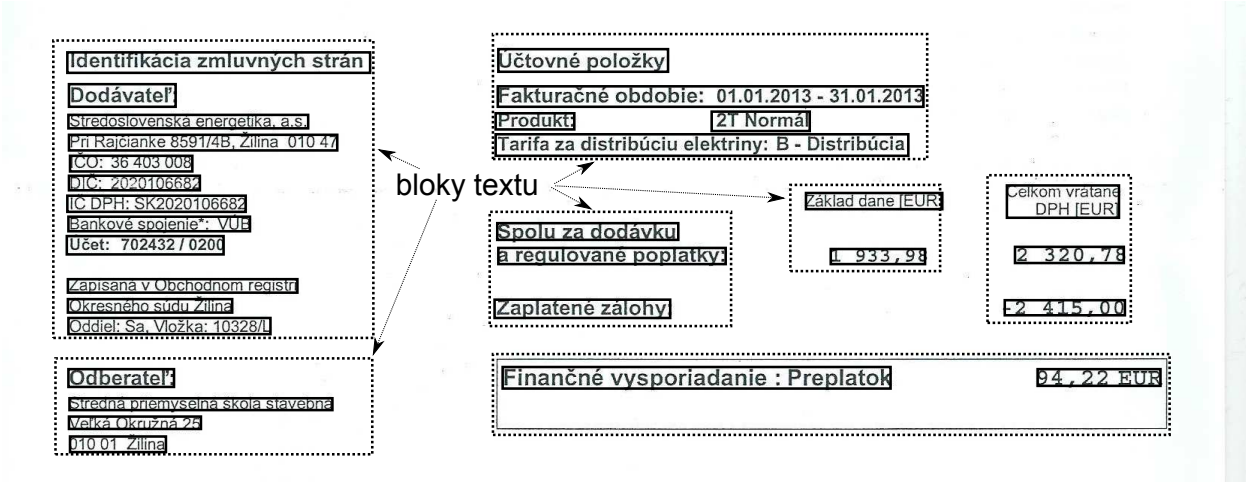
Hlavným účelom tohoto dokumentu je extrahovať z dokumentu text spolu s jeho súradnicami. Súradnice sa priraďujú jednotlivým slovám, riadkom a celkovému dokumentu.



Obrázek 3.2: Príklad SpatialWord

Označenie **SpatialWord** definuje slovo, ktoré má priradené súradnice. Súradnicami slova sa sa myslia dva body, ktoré obsahujú súradnice X a Y. Na obrázku 3.2 sú zobrazené dve slová *SpatialWord*. Hranice slova sú definované bodmi *leftTop* a *rightDown*, kde každý má svoju X-ovu a Y-ovú súradnicu. V tomto súradnicovom systéme sa bod so súradnicami (0,0) nachádza v ľavom hornom rohu dokumentu. Jedná sa teda o klasický súradnicový systém používaný v grafických prostrediach.

Označenie **SpatialLine** definuje skupinu susediacich slov, medzi ktorými je malá medzera. Táto medzera by nemala presahovať dvojnásobok výšky písma.



Obrázek 3.3: Príklad SpatialLines

Vo väčšine prípadov totiž väčšia medzera oddeľuje slová z dvoch rôznych blokov textu. Blok textu je abstrakciou, ktorá priraďuje skupine prvkov *SpatialLine* význam, ako napríklad adresa odoberateľa, tabulka, kontakté údaje a tak podobne.

Slová *SpatiaWord*, ktoré tvoria *SpatialLine* sú obalené najmenším možným obdĺžnikom (ako na obrázku **SpatialWord**). Súradnice ľavého horného a pravého dolného bodu tohoto obdĺžnika tvoria súradnice *SpatialLine* (rovnako ako u *SpatialWord*). Obrázok . Ďalším účelom *SpatialLine* je rozdeľovať slová do blokov textu, teda že jedna *SpatialLine* patrí len jednému bloku textu.

Výsledný dokument, označovaný ako *SpatialDocument* súradnice nemá. Má len šírku a výšku dokumentu, pretože súradnice ľavého horného bodu sú vždy *(0,0)*.

Pre prevod z rôznych typov dokumentov (txt, pdf, jpg, png alebo bmp) sú použité rôzne moduly. Napríklad pre extrakciu textu z obrázku sa používa OCR[1] modul, pre PDF dokument sa použije modul pre prevod PDF do obrázku, ktorý je následne poslaný do OCR modulu[2], alebo textový súbor, je odoslaný do jednoduchého modulu, ktorý len pridá informáciu o umiestnení slov, či riadkov v dokumente.

3.2.1 Prevod z textového súboru

Modulu pre prevod textového súboru do *SpatialDocument*u má niekoľko dôvodov.

- takmer každý dokument (rtf, doc, docx, xml) sa ľahšie prevádza do textového súboru než do obrázku
- väčšina OCR aplikácií umožňuje prevod obrázku do textového súboru, čo dáva možnosť používať vlastné OCR namiesto *Tesseract OCR*, ktorý je zabudovaný do aplikácie.

[1]OCR je zkratka Optical character recognition. Je to metóda, ktorá z obrázku extrahuje text

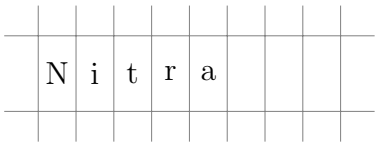
[2]existujú knižnice, ktoré dokážu extrahovať text s PDF dokumentu, ale nedokážu to ak je tvorený výhradne obrázkami. V tom prípade je opäť potrebné extrahovať obrázky do súborov, nad ktorými sa následne spustí OCR aplikácia

- vytvorený textový súbor sa veľmi rýchlo prevádza do *SpatialDocument*u, čo pri vývoji a testovaní aplikácie ušetrilo veľa času

Modul rozdelí celý obsah textového dokumentu na jednotlivé slová a riadky, ktorým následne priradí súradnice, čím z nich vytvorí *SpatialWord* a *SpatialLine*.

Vytvorenie SpatialWord a SpatialLine

Ak by bol celý textový dokument rozdelený po znakoch na tabuľku, kde jedena bunka tabuľky obsahuje jeden znak textového súboru, potom súradnice slova je sú určené riadkom a stĺpcami tabuľky, v ktorých sa nachádzajú jednotlivé znaky slova. Príklad slova je ilustrovaný na obrázku 3.4, kde slovo je tvorené piatimi znakmi v piatich bunkách tabuľky.



Obrázek 3.4: Lokácia slova

Nech každý znak (a teda každá bunka tabulky) je 24 pixelov vysoký a 12 pixelov široký [3], potom dokument je možne virtuálne previesť do tlačenej podoby a slovám dať adekvátnejšie súradnice. Samozrejme tieto rozmery je možné zmeniť ľubovolne, pretože na funkčnosť to nebude mať vplyv, ale dať znakom takýto rozmer je lepšie pre predstavu, než nechávať znaky (respektíve bunky tabuľky) o veľkosti 1x1 pixel.

Slová *SpatialWord* sa teda jednoducho vytvoria tak, celý dokument je prehľadávaný po riadkoch a tie po slovách. Každému slovu je vytvorené *SpatialWord* so súradnicami vypočítanými podľa vzorca v kóde 3.1, kde *wordStart* je pozícia, kde slovo začína a *wordSize* je počet znakov v slove:

Kód 3.1: Zmiešaná definícia

```
1  leftTop.x = (wordStart) * 12
2  leftTop.y = row * 24
3  rightDown.x = (wordStart+wordSize) * 12
4  rightDown.y = (row+1) * 24
```

Tie *SpatialWord*, ktoré majú medzi sebou medzeru menšiu ako 3 bunky, sa spoja do jedného riadku *SpatialLine*.

3.2.2 Prevod PDF na obrázok

PDF dokument môže obsahovať text, ktorý je jednoducho extrahovateľny[4]. Takýto dokument je ale vo väčšine generovaný zo šablóny a k užívateľovi bol poslaný elektronickou cestou. Z tekéhoto dokumentu by sa teda dalo jednoducho vyextrahovať

[3] rozlíšenie znaku 12x24 pixelov je často používaným v ihličkových, alebo termo-tlačiarniach, preto práve tento rozmer

[4] text sa dokonca dá extrahovat v aplikácií Adobe Reader jedoduchým označením a zkopírovaním do Clip Boardu

text a s jeho lokáciou, ale PDF dokumenty, ktoré sú výstupom zo skenera, už text obsahovať nemusia. V prípadoch, keď PDF dokument obsahuje text[5], je kvalita rozpoznaného textu nízka. Preto pre zvýšenie kvality rozoznania textu sa PDF dokument prevedie po stránkach do obrázkov a tie sú následne odoslané do OCR modulu.

Pre prevod PDF dokumentov bola v tejto práci použitá knižnica *Pdf-renderer,* ktorá je podprojektom projektu *Swinglabs*[6].

3.2.3 OCR modul

OCR modul má za úlohu zo vstupných obrázkov vyextrahovať text a určiť jeho umiestnenie v dokumente. Pre extrakciu textu z obrázka, bolo ako najlepšie riešenie zvolený open-source engine Teseract OCR. Okrem podpory viacerých jazykov je možné naučiť Tesseract OCR tiež nove znakové sady a hlavne pri extrahovanom texte tiež uvádza súradnice v obrázku, odkiaľ text pochádza. Výstupom tohoto modulu je *SpatialDocument*, ktorý pozostáva z *SpatiLine* a tie z *SpatialWord*.

3.2.3.1 Predspracovanie pred OCR

Keďže v praxi nie sú všetky dokumenty dokonale oskenované, dochádza najčastejšie k pootočeniu dokumentu. Tesseract nie je úplne spoľahlivý a sami autori doporučujú, aby bol obrázok predspracovaný, tj. aby bol dokument vyrovnaný, aby bol odfiltrovaný šum a tak podobne. Pokusy naviac ukázali, že keď tesseract zpracováva menšie časti obrázku (napríklad pár riadkov), má lepšie výsledky než pri rozoznávaní celého dokumentu. Táto časť diplomovej práce sa preto zameriava aj na to, ako najviac pomôcť Tesseractu pri extrahovaní textu z obrázkov. Venuje vyrovnaniu obrázku (takzvane *descew* obrázku), rozsekaniu na menšie časti a následnému zloženiu obrázku.

3.2.3.2 Vyrovnanie obrázku

Oskenované dokumenty bývajú často pootočené o pár stupňov. Pre vyrovnanie obrázku bola použitá Houghova transformácia [1]. Základná Houghova transformácia bola určená pre detekciu priamok v obrázku. Rozšírená verzia dokáže na obrázku detekovať ľubovolné tvary, ako kruhy alebo elipsy. Pre účely tejto práce úplne postačila jednoduchá verzia. Pri implementovaní bola použitá existujúca implementácia [2] vyvinutá jazyku Java, ktorej autor je ale neznámy (bol uvedený len jeho pesudonym *Anydoby*). Implementácia vyrovnania obrázku pozostáva z dvoch fází. V prvej sa na obrázku detekuje uhol, o ktorý je oskenovaný dokument pootočený a v druhej fáze je pôvodný obrázok otočený naspäť o zistený uhol. Centrom rotácie je samozrejme stred obrázku.

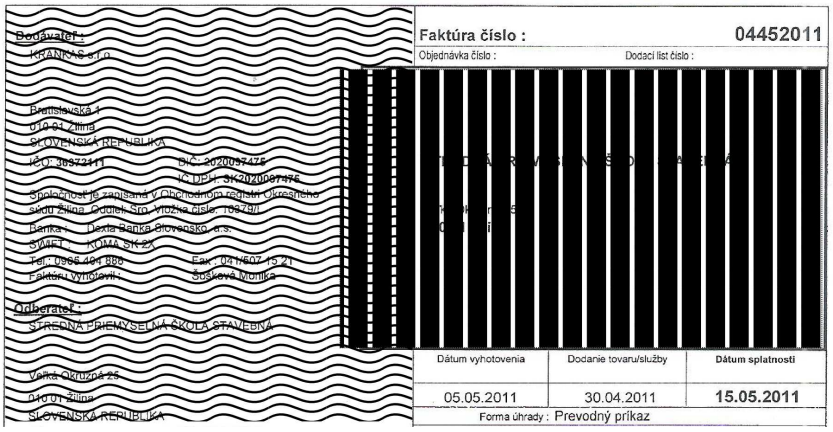
[5]text získaný pomocou OCR v skeneri, alebo OCR v obslužnej aplikácií skeneru
[6]https://java.net/projects/pdf-renderer

3.2.3.3 Rozsekanie obrázku

Tesseract OCR má problém pri extrahovaní textu z veľkých obrázkov. Pri pokusoch, vracal oveľa lepšie výsledky v prípadoch, keď extrahoval text len z častí obrázku. Tieto boli ručne vybrané a uložené do separátneho súbora. Tieto časti väčšinou obsahovali bloky textu.

Pre segmentáciu dokumentu existuje mnoho spôsobov. Medzi najčastejšie používané algoritmy patria X-Y cutting, Voronoi, Whitespace alebo Docstrum. V práci [3] sú tieto algoritmy porovnávané. Ikeď Voronoi a Docstrum dávajú dobré výsledky a dokážu spracovať aj zakrivené dokumenty, pre účely rozsekania obrázku najlepšie poslúži X-Y cutting altorgitmus v kombinácii s detekovaním hrán.

Výsledkom X-Y cutting algortmu sú obdĺžniky, ktoré sa dajú z obrázka jednoducho vybrať do samostatného súboru. Iné algoritmy môžu mať za výsledok výrezy v tvare písmena L, s čím sa oveľa horšie pracuje, pretože pri vyrezávaní (kde sa vyrezáva obdĺžnik) je nutné prekryť oblasti z iných výrezov. Príklad je uvedený na obrázku 3.5, kde je problémová oblasť je označené vlnkami. Do nej zasahuje pruhovaná oblasť a pri výreze by ju bolo nutné prekresliť, aby sa nezpracovávala *Tesseract*om. Detekcia hrán v dokumente nám umožní lepšie detekovať *biele miesta* a zároveň umožní detekciu čiar, ktoré môžme použiť ako oddeľovače blokov textu.

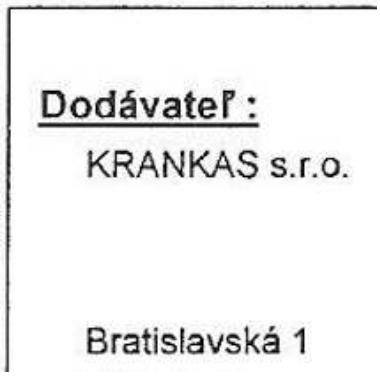


Obrázek 3.5: Problém s vyrezávaním

Detekcia hrán

Pre detekovanie hrany sme použili Canny Edge detector[4]. Algortmus väčšinou pozostáva zo 4. krokov:

- Eliminácia šumu pomocou Gaussového filtra
- Určenie gradientu (veľkosť a smer) z prvej derivácie
- Nájdenie lokálnych maxím



Obrázek 3.6: Originánly obrázok

Obrázek 3.7: Detekované hrany

- eliminácia nevýznamných hrán pomocou dvojitého tresholdu (s hornou a dolnou hranicou)

Na obrázku 3.6 je zobrazený pôvodný dokument. Jeho detekované hrany sú zobrazené na obrázku 3.7.

Jeho implementáciu vytvoril Tom Gibara[5], ktorá bola s menšími úpravami použitá v tejto práci. Bolo nutné pridať podporu formátu obrázku 4-bytového ABRG. Pre dvojitý treshlold bola nastavená dolná hranica na 0.9 (spodný treshold) a hornú na 1.0 (horný treshold)[7]. Táto kombinácia pri testovaní na oskenovaných dokumentoch vracala najlepšie výsledky.

Detekcia deliacich čiar

Deliacimi čiarami sa označujú horizontálne alebo vertikálne čiary, ktoré dokument rozdeľujú do blokov. Je ale dôležité, aby nenarušili súvislosť textu. Napríklad aby nerozdeľovali slovo, alebo jeden riadok na dve časti.

Deliace čiary vychádzajú z dvoch zdrojov. Jedným zdrojom sú čiary priamo vytlačené v dokumente, ktoré sa označujú ako **Pritned Dividers**. Väčšinou tvoria tabuľky alebo horizontálne a vertikálne čiary, ktoré oddeľujú časti dokumentu (napríklad hlavičku dokumentu od zbytku).

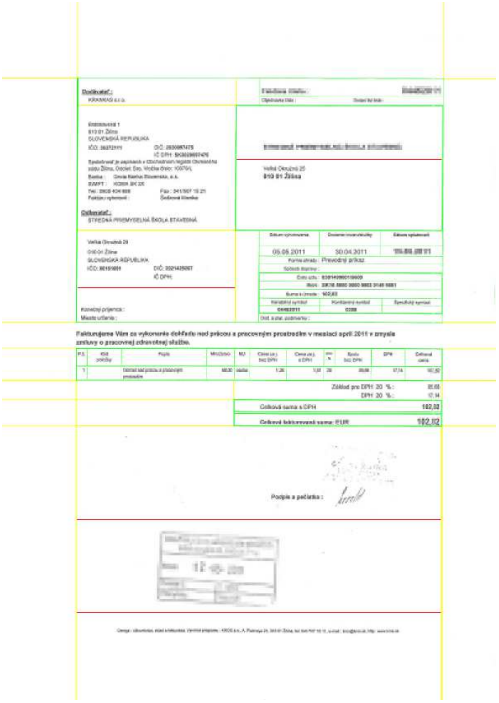
Predĺžením *Pritned Divider*, až po okraj dokumentu, alebo po najbližšiu *Pritned Divider*, by vznikla ďalšia deliaca čiara. Takéto deliace čiary sa označujú ako **Extended Printed Divider**. Podmienkou je, aby žiadna *Extended Printed Divider* nekrižovala inú *Pritned Divider* alebo text.

Druhým zdrojom deliacich čiar sú biele miesta v dokumente. Biele miesta dokumente tvoria prázdnu oblasť dokumentu, kde nie je nič "vytlačené". Deliace čiary, ktoré vychádzajú z bielych miest v dokumente, budú označené ako **White Space Divider**.

Na obrázku 3.8 je zobrazený dokument. Obsahuje tabuľky, polo-štrukturovaný text, ktorý je naviac orámovaný. Obrázok 3.9 ilustruje, ako by takýto dokument

[7]Ak hodnota daného pixelu je väčšia ako horný treshold, je pixel označený priamo ako hranový. Ak je graditent medzi horným a dolným tresholdom, potom je uznaný ako hranový, pokiaľ susedí s bodom, ktorý už je označený ako hranový.

Obrázek 3.8: Dokument



Obrázek 3.9: Dokument a jeho deliace čiary

mohol byť rozdelený podľa deliacich čiar. Zelené čiary predstavujú *Pritned Dividers*. Ich predĺžením sme dostali *Extended Printed Dividers*, ktoré sú zobrazené žltou farbou. *White Space Dividers* sú zobrazené červenou.

Pri hľadaní deliacich čiar budeme v prvom kroku hľadať *Pritned Dividers*.Takéto čiary ale nie vždy končia na okraji. Preto v druhom kroku sa pokúsime nájdené hrany predĺžiť skrz biele miesta na maximum, čím dostaneme *Extended Printed Dividers*. V treťom kroku sa budeme snažiť rozdeliť vzniknuté obdĺžniky skrz ich biele miesta, čím vytvoríme *White Space Dividers*.

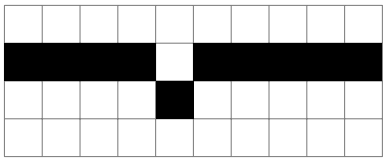
Detekcia čiar na obrázku

Obrázok s detekovanými hranami obsahuje len biele a čierne body. Biele body predstavujú pozadie (respektíve oblasť, kde sa farba nemení), čierne nájdené zmeny farby, to jest miesta, kde sa mení farba pozadia a farba textu/čiar[8]. Tento obrázok bude v tejto práci používaný ako základný obrázok pre detekciu všetkých deliacich čiar a označuje sa ako **Základný obrázok**.

Algoritmus pre nájdenie *Pritned Dividers* prehľadáva všetky čierne body v *základnom obrázku* a skúša, akú najdlhšiu horizontálnu a vertikálnu čiaru je možne z daného bodu nájsť. Pretože obrázok nie je dokonalý a obsahuje ne-

[8]V zdrojových kódoch je to opačne. Čierny pixel predstavuje bod, kde sa farba nemení, a biely, resp. nečierny bod označuje zmenu farby. Pretože biele miesto v pôvodnom obrázku a čierne miesto v obrázku s rozoznanými hranami je tá istá oblasť, bolo by pri výklade mätúce používať obe tieto označenia naraz. Preto pre potreby tejto práce boli v obrázku s rozoznanými hranami prehodené biely za čierny pixel, aby *biela oblasť* bolo spoločné označenie v oboch obrázkoch.

rovnosti, je potrebné predpokladať, že rovná čiara ma občas posunutý pixel, ako napríklad na obrázku 3.10.



Obrázek 3.10: Nerovnosť hrany

Preto pri hľadaní horizontálnej čiary zľava doprava, najprv algoritmus skontroluje, či od aktuálneho pixelu je vpravo tiež čierny bod. Ak áno, pridá ho do aktuálnej horizontálnej čiary a pokračuje, inak skontroluje diagonálne (v smere doprava) susediace body a ak aspoň jeden je čierny, pridá ho do aktuálnej horizontálnej čiary a pokračuje v hľadaní. Priorita pixelu označuje susedný pixel, ktorý pri prehľadávaní horizontálnej/vertikálnej čiary má prednosť. Priority susedných pixelov sú zobrazené na obrázkoch 3.11 a 3.12. Pri hľadaní vertikálnej čiary postupuje obdobne ako pri hľadaní horizontálnej čiary s tým, že sa prehľadáva zhora nadol.

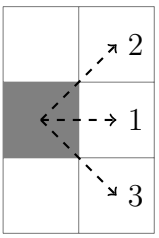
Hrany, ktoré majú dĺžku 1 pixel sú odstránené. Obrázok 3.13 zobrazuje detekované hrany, ktoré sme rozdelili na vertikálne a horizontálne. Na obrázku 3.14 sú nájdené horizontálne hrany zobrazené červenou a vertikálne zelenou farbou. Žlté pixely zobrazujú prieniky vertikálnych a horizontálnych hrán.

Je vidieť, že aj znaky textu obsahujú horizontálne a vertikálne čiary. Pre rozlíšenie, ktoré hrany sú deliacimi a ktoré sú čiarami v znakoch je použitá minimálna dĺžka v danom smere. Minimálna dĺžka v horizontálnom smere je nastavená na 1/10 z šírky dokumentu a minimálna dĺžka vo vertikálnom je 1/10 z výšky dokumentu. Predpokladom je, že dokument nebude obsahovať znak, ktorý by zaberal priestor aspoň 1/10 šírky, alebo výšky dokumentu. Na druhú stranu čiary, ktoré by boli kratšie ako 1/10 šírky, alebo výšky, zrejme oddeľujú veľmi malý blok textu.

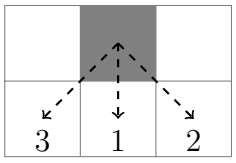
Na obrázkoch 3.6 a 3.7 je vidieť, že *Printed Dividers* sa po detekcií hrán prejavujú ako dve čiary [9], respektíve ako krivka, ktorá obaľuje pôvodnú čiaru. Tieto čiary je potrebné zjednotiť, aby sa dokument nerozdeľoval podla dvoch, tesno susediacich čiar, ktoré by navyše v ďalších krokoch[10] mohli spôsobiť problémy.

[9] detekcia hrán hľadá, kde sa mení farba, takže každá čierna krivka a čiara je teda ohraničená z oboch strán

[10] Napríklad pri predlžovaní čiar, keď sa dve kolmé čiary križujú v tvare ⊫ a horizontálna čiara



Obrázek 3.11: Poradie pri hľadaní ďalšieho pixelu pre horizontálnu čiaru



Obrázek 3.12: Poradie pri hľadaní ďalšieho pixelu pre vertikálnu čiaru

Obrázek 3.13: Detekované hrany

Obrázek 3.14: Detekované hrany rozdelené na horizontálne a vertikálne

Získanie čiar *Printed Dividers* pozostáva z niekoľkých krokov:

1. prefiltrovanie hrán: hranu obalíme do čo najmenšieho obdĺžnika. Obdĺžnik musí byť široký minimálne 1/10 šírky dokumentu, alebo vysoký 1/10 z výšky dokumentu, pričom menší z pomerov šírka:výška a výška:šírka nesmie má byť menší ako 1:10 [11].

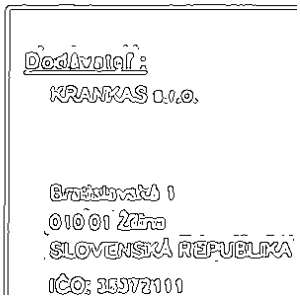
2. roztriedenie hrán do dvoch množín: na vertikálne a horizontálne

3. zjednotenie susediacich hrán: Zvlášť pre vertikálne a zvlášť pre horizontálne čiary. Zjednotenie vertikálnych čiar sa spočíta (pre horizontálne čiary obdobne):

   - zoradenie čiar podľa x-ovej súradnice
   - meranie vzdialeností: postupne pre každú vertikálnu čiaru sa preskúmajú, v smere doprava, susedné vertikálne čiary (maximálne do vzdialenosti 1/10 šírky dokumentu), ktoré prekrývajú vyšetrovanú čiaru vo vertikálnom smere. Namerané horizontálne vzdialenosti sa uložia. Vzdialenosť dvoch čiar sa počíta ako vzdialenosť obdĺžnikov, ktoré čiary obaľujú.
   - z nameraných vzdialeností sa zistí maximálna používaná vzdialenosť, ktorá sa označuje ako **Susedská vzdialenosť**. V tomto prípade by ako maximálna vzdialenosť bola braná tá, ktorá by bola aspoň 3x zaznamenaná.
   - zjednotenie susedných čiar podľa nájdenej maximálnej vzdialenosti: dve susediace čiary, ktoré majú horizontálnu vzdialenosť menšiu alebo

by mohla byť predĺžená. V prípade, keby bola vertikálna čiara ponechaná ako dve, vonkajšia čiara (v tomto prípade tá vľavo) by zastavila pokus o predĺženie horizontálnych čiar smerom doľava.

[11] pomer 1:10 odfiltruje hrany s väčším sklonom a ponecháva hrany, ktoré sú mierne zakrivené, ako napríklad hrany, zaobleného obdĺžnika, ktoré sú na koncoch zakrivené



Obrázek 3.15: Pôvodné hrany

Obrázek 3.16: Zjednotené hrany

rovnú ako *Susedská vzdialenosť* (tj. vzdialenosť obdĺžnikov, ktoré čiary obaľujú), sa spoja do jednej čiary: čiaru obaľujúci obdĺžnik je spočítaný ako minimálny obdĺžnik, ktorý obaľuje obe čiary (tj. z ich obdĺžnikov), z neho sa vytvoria body, ktoré vedú stredom obdĺžnika od horného okraju až po spodný okraj.

4. nahradenie pôvodných čiar zjednotenými čiarami: z obrázku sa odstránia body, ktoré tvorili pôvodnú čiaru, a nahradia ich body, ktoré tvoria zjednotené čiary.

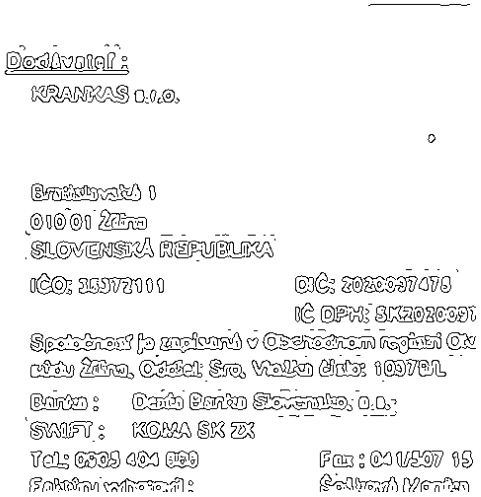
Na obrázku 3.15 a 3.16 sú zobrazené čiary pred zjednotením a po zjednotení.

Predĺžením takto nájdených čiar by bolo možne dostať adekvátne rozdelenie dokumentu na menšie bloky. Predĺženie v danom smere prebieha po bielych miestach dokumentu, dokiaľ nenarazí buď na znak, alebo na inú deliacu čiaru. Takáto čiara ale môže preseknúť jedno slovo na dve časti, pretože medzi jednotlivými znakmi sú biele miesta. To je samozrejme nechcené a preto ešte pred predlžovaním čiar je nutné vyplniť prázdne miesta medzi znakmi jedného slova.

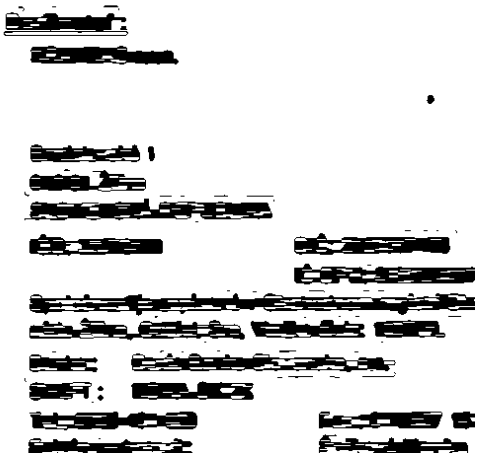
Základný obrázok sa preto prehľadáva pixel po pixeli a počítajú sa dĺžky bielych horizontálnych čiar, ktoré sú menšie ako 1/10 šírky dokumentu. Najčastejšia sa opakujúca dĺžka určuje najbežnejšiu vzdialenosť medzi čiernymi pixelmi, čo je v pôvodnom dokumente predstavuje šírku čiary znakov. Táto hodnotu sa zväčší o 2 pixely [12] a označí ako **Hrúbka písma**.

Pre zistenie najčastejšej vzdialenosti medzi znakmi je potrebné zistiť druhú najbežnejšiu vzdialenosť. Preto sa z nameraných vzdialeností odstránia tie, ktoré sú menšie ako *Hrúbky písma* a z týchto hodnôt sa vypočíta priemer. 1,5 násobok tejto hodnoty (pri pokusoch sa ukázalo, že 1,5 násobok spočítanej strednej hodnoty pokrýva väčšinu medzier medzi znakmi) bude označovaný ako **Znaková medzera**.

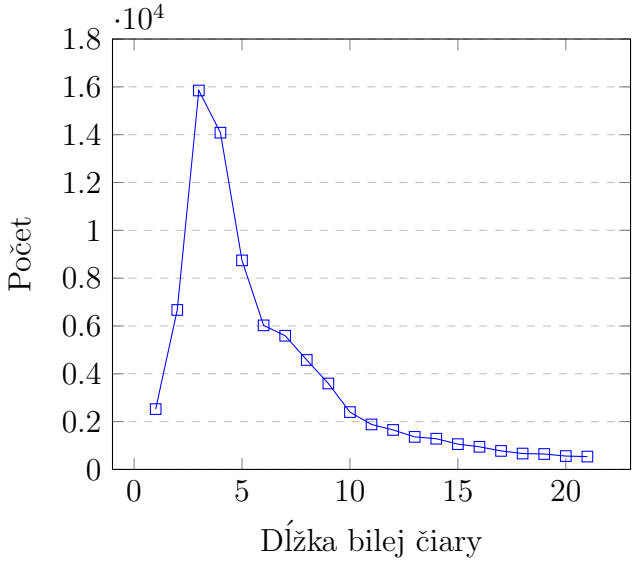
[12] predĺženie o 2 pixle sa pri pokusoch ukázalo ako dostatočné, aby bola obsiahnutá väčšina medzier v znakoch



Obrázek 3.17: Pôvodné hrany znakov



Obrázek 3.18: Vyplnenie medzier medzi znakmi



Základný obrázok bude opäť postupne prechádzaný pixel po pixeli a všetky nájdené horizontálne biele čiary, ktoré sú kratšie ako *Znaková medzera*, sa nahradia čiernymi pixelmi (ďalšie spracovanie bude tým pádom pokladať tieto pixely ako pixely patriace znaku). Tým sa vyplnia biele miesta nie len v znakoch, ale aj medzi jednotlivými znakmi z jedného slova. Obrázok 3.17 zobrazuje obrázok pred a obrázok 3.18 po vyplnení medzier medzi znakmi.

Predĺženie nájdených čiar

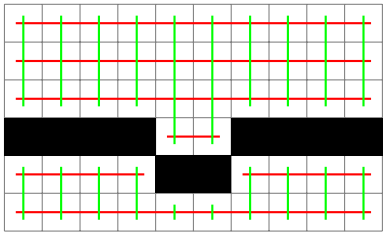
Predĺžením doposiaľ nájdených rozdeľovacích čiar *Printed Dividers* vznikajú *Extended Printed Dividers*, ktoré adekvátnejšie rozdeľujú dokument než čiary *White Space Divider*. Príkladom môže byť horizontálna čiara, ktorá končí 3 cm od okraja dokumentu (tj. nepretína dokument úplne), alebo tabuľka s neviditeľnými hranami, ktorá má orámovaný len posledný riadok.

Čiara *Printed Divider* predlžovaná dovtedy, pokiaľ algoritmus nenarazí na

okraj dokumentu, alebo na *Printed Divider* čiaru. Čiara nie je predĺžená, ak by v danom smere pretínala znak. Môže ale pretínať iné *Extended Printed Dividers*, ako je vidieť na obrázku 3.9, kde *Extended Printed Dividers* sú zobrazené žltou farbou.

Nájdenie rozdeľovačov v bielych miestach

Zo všetkých bielych pixelov v *Základnom obrázku* sa vytvoria vertikálne a horizontálne biele čiary. Teda v každom stĺpci bude toľko vertikálnych bielych čiar, koľko spojitých úsekov z bielych pixelov sa v ňom nachádza. Obdobne s riadkami a horizontálnymi čiarami. Príkladom je obrázok 3.19, kde sa nachádza 7 horizontálnych a 20 vertikálnych bielych čiar. Pre úsporu pamäte a ušetrenia výpočtu v následujúcich krokoch, odfiltruje z horizontálnych tie, ktoré sú kratšie ako 1/10 šírky dokumentu a z vertikálnych tie, ktoré sú kratšie ako 1/10 výšky dokumentu.



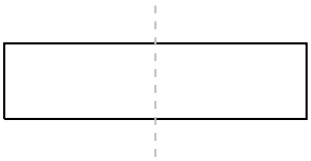
Obrázek 3.19: Čiary v bielych miestach

Rozsekanie podľa deliacich čiar

Dokument sa rekurzívne rozseká na menšie bloky pomocou získaných deliacich čiar *Printed Dividers*, *Extended Printed Dividers* a *White Space Divider*. Najprv sa z nich vyberie jeden kandidáta (čiaru) pre čo najlepšie rozdelenie bloku. Táto čiara je označovaná ako čiara **Divider Line**, rozdelí blok na dva a delí tiež všetky deliace čiary patriace bloku. Pri delení sa opäť odfiltrujú tie čiary, ktoré sú kratšie ako 1/10 rozmeru dokumentu. Rozmer 1/10 dokumentu udáva rozdelenie jedného dokumentu na maximálne 100 blokov. Pri hľadaní kandidáta z deliacich čiar má väčšiu prioritu typ čiary než to, či sú horizontálne, alebo vertikálne. Vyhľadáva sa najprv v čiarach *Printed Dividers* (označené skratkou PD), potom *Extended Printed Dividers*(označené skratkou ED) a nakoniec *White Space Divider*(označené skratkou WD).

Ak čiary, v ktorých sa hľadá *Divider Line*, sú horizontálne budú sa značiť s prefixom H, ak vertikálne, tak s prefixom V. Napríklad horizontálne čiary *Extended Printed Divider* sa budú označovať ako HED (ak by boli vertikálne, tak VED).

Druhou prioritou, ako by sa mal blok deliť, je rozmer bloku. V prípade, ak je blok širší než vyšší, mal by sa rozseknúť vertikálne, ako je na obrázku 3.20. Naopak ak je vyšší než širší, mal by sa rozseknúť horizontálne, ako na obrázku 3.21.



Obrázek 3.20: Vertikálne rozdelenie

Obrázek 3.21: Horizontálne rozdelenie

Dodávateľ :
KRANKAS s.r.o.

Bratislavská 1
010 01 Žilina
SLOVENSKÁ REPUBLIKA
IČO: **36372111** DIČ: **2020097475**
IČ DPH: **SK2020097475**
Spoločnosť je zapísaná v Obchodnom registri Okresného súdu Žilina, Oddiel: Sro, Vložka číslo: 10379/L
Banka : Dexia Banka Slovensko, a.s.
SWIFT : KOMA SK 2X
Tel.: 0905 404 888 Fax : 041/507 15 21
Faktúru vyhotovil : Šošková Monika

Odberateľ :
STREDNÁ PRIEMYSELNÁ ŠKOLA STAVEBNÁ

Obrázek 3.22: Pôvodný obrázok

Dodávateľ :
KRANKAS s.r.o.

Bratislavská 1
010 01 Žilina
SLOVENSKÁ REPUBLIKA
IČO: **36372111** DIČ: **2020097475**
IČ DPH: **SK2020097475**
Spoločnosť je zapísaná v Obchodnom registri Okresného súdu Žilina, Oddiel: Sro, Vložka číslo: 10379/L
Banka : Dexia Banka Slovensko, a.s.
SWIFT : KOMA SK 2X
Tel.: 0905 404 888 Fax : 041/507 15 21
Faktúru vyhotovil : Šošková Monika

Odberateľ :
STREDNÁ PRIEMYSELNÁ ŠKOLA STAVEBNÁ

Obrázek 3.23: Nájdené rozdelenie obrázka

Preto ak je blok širší, bude sa *Divider Line* hľadať v deliacich čiarach v tomto poradí: VPD, HPD, VED, HED, VWD, HWD. Ak je blok vyšší, tak v tomto poradí : HPD, VPD, HED, VED, HWD, VWD.

Pri prehľadávaní každej skupiny má väčšiu prioritu tá čiara, ktorá je najbližšie stredu:

1. všetky čiary sa zoradia podľa vzdialenosti od stredu bloku (tj. čiara, ktorá by viedla stredom, by bola prvá)

2. ak výsledný zoznam nie je prázdny, vyberie sa prvá a tá sa stane čiarou *Divider line*

Ak v danom bloku bola nájdená *Divider line*, rozdelíme ho podľa nej na dva a zároveň aj všetky ich deliace čiary (tj. PD, ED a WD) . Ak sú vzniknuté bloky dostatočne veľké a obsahujú aspoň nejakú deliacu čiaru, rekurzívne sa rozdelia podľa ich deliacich čiar (tj. nájsť v nich *Divider line* a podľa nich ich rozdeliť).

Obrázok 3.22 ukazuje, ako vyzeral pôvodný dokument a obrázok 3.23 zobrazuje, ako bol dokument rozdelený. Je vidieť, že deliace čiary sa pridržujú tých vytlačených, a čo je dôležité, nerozdeľujú slová na dve časti.

3.2.3.4 Výsledné poskladanie

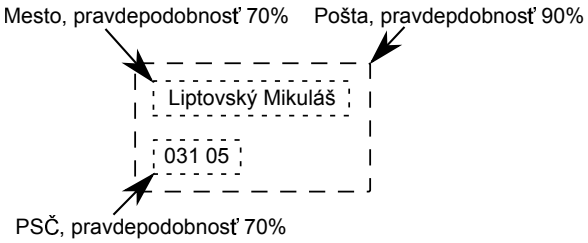
Časti obrázku, na ktoré bol dokument rozsekaný, sa uložia ako jednotlivé obrázkové súbory, nad ktorými je spustený Tesseract OCR. Jeho výsledkom sú rozoznané slová s ich pozíciou. Poskladanie výsledného dokumentu (jeho extrahovaných slov z pôvodného obrázku) je jednoduché, pretože ku každému obrázkovému súboru sa udržiava informácia, z ktorej pozície bol obrázok vyrezaný.

3.3 Anotátor dokumentov

Základným prvkom, s ktorým pracuje anotátor dokumentov je **fráza**. Tvoria ju slová, ktoré sa môžu nachádzať na rôznych miestach v dokumente. Väčšinou sa ale jedná o prípady, kedy sa slová nachádzajú vedľa seba, alebo pod sebou. Anotátor podľa užívateľom definovaných pravidiel nachádza v dokumente fráze a následne im priraďuje význam a pravdepodobnosť, že daná fráza skutočne odpovedá priradenému významu. Spôsob, ako užívateľ definuje pravidlá a ako sa počíta pridelená pravdepodobnosť je popísané v kapitole Definícia anotácie.

Príkladom môže byť číslo *031 05*. To podľa užívateľom nadefinovaného pravidla, by malo byť *PSČ* s pravdepodobnosťou 20% . Číslo ale môže byť súčasťou telefónneho čísla napríklad *00421 55 031 05*. To, či je číslo skutočne *PSČ* závisí od text v jeho okolí. Ak by sa naľavo, alebo nad číslom *031 05* nachádzalo mesto, pravdepodobnosť, že by sa jednalo skutočne o *PSČ*, by bola väčšia. To ale závisí od nadefinovaných pravidiel.

Fráza, ktorá má priradený význam a pravdepodobnosť sa označuje ako **anotovaná fráza**. Obrázok 3.24 zobrazuje hneď niekoľko anotovaných frází. Jednou je *Liptovský Mikuláš*, ktorá bola anotovaná ako typ *Mesto* s pravdepodobnosťou 70%. Druhou je číslo *031 05*, ktorá bola anotovaná ako typ *PSČ* s pravdepodobnosťou 70%. Tretia fráza sa skladá z týchto dvoch fáz, je označená ako typ *Pošta* s pravedpodobnosťou 90%.



Obrázek 3.24: Anotované fráze

V súčasnosti existuje mnoho anotačných nástrojov, ale väčšina sa sústreďuje na anotáciu textu bežného jazyka a prevod textov do RDF dát[6], alebo sú súčasťou komerčných produktov. Preto bolo nutné vyvinúť vlastný engine pre anotovanie polo-štrukturovaných dokumentov.

***Polo-štrukturovaný dokument**, alebo dáta, je forma štrukturovaných údajov, ktoré nie sú v súlade s formálnou štruktúrou dátových modelov, spojených s relačnou databázou, alebo s inou formou dátových tabuliek, ale napriek tomu obsahujú značky, alebo iné prvky, ktoré vytvárajú hierarchiu záznamov a dátových polí*[7].

Zjednodušene sú to dáta, ktorých polia (v angličntine *fields*, v tejto práci označované ako *fráze*) sú popísané nejakou značkou. Príkladom je *"tel.: +420 111 222 445"*, kde fráza je telefónne číslo a značkou je *"tel:"*. Značkou môže ale byť aj poloha fráze v dokumente, respektíve to, vedľa akých ďalších fráz sa nachádza.

3.3.1 Anotácia fráz

Anotácia frázy znamená priradenie významu jednému, alebo viacerým slovám, ktoré spolu nejakým spôsobom súvisia. Význam fráze sa môže meniť v závislosti na kontexte, umiestnení alebo jeho zvýraznení [13]. Automatickému anotovaniu fráz a textu sa venuje celé vedecké odvetvie a je veľmi komplikované.

V tejto práci sa využíva toho, že dáta sú polo-štrukturované a kontext nie je potrebné pochopiť úplne. Navyše v takýchto dokumentoch má každá významná fráza svoj pomerne jednoducho definovateľný kontext, alebo pre určenie významu fráze nie je potrebný kontext.

Anotovanie fráz v tejto práci, je založené na pravidlách, ktoré majú v sebe 3 základné položky:

- *pattern* - jednoznačne definovaný výraz pre nájdenie slova/skupiny slov
- *type,probability* - priradenie typu a pravdepodobnosti nájdenému slovu/skupine slov
- *context*- dodatočné zvýšenie pravdepodobnosti v prípade, že sa v okolí slova/slov nachádza fráza konkrétneho typu

Konkrétnym definíciám a príkladom sa venuje kapitola Definícia anotácie.

V implementácii sa ako *pattern* používajú:

- konštanty - môžu byť menovito definované, alebo ako stĺpec v tabuľke. Naviac je možné definovať či porovnávanie je *Case Sensitive*[14], *Case Insensitive*[15], alebo len veľkosť prvého znaku musí byť v oboch prípadoch zhodná[16].
- regulárne výrazy
- nájdené fráze - používa sa pri komplexných anotáciách, kde výsledná fráza pozostáva s niekoľkých anotovaných fráz[17].

Context definuje podmienku: ak sa v nejakom okolí od aktuálnej fráze nachádza fráza typu X, potom zvýš pravdepodobnosť aktuálnej fráze o Y. Viac o contexte je popísané v sekcii Kontext.

O nájdenie výrazov v textovom dokumente stará trieda **Annotator**.

3.3.2 Definícia anotácie

V súbore *config/annotate.xml* je možné definovať vlastné anotovanie fráz. Tento súbor sa označuje ako **Anotačný súbor**. Definície anotácií sa delia na dva typy: **Jednoduchá anotácia** a **Komplexná anotácia**. Princíp anotovania je postavený na:

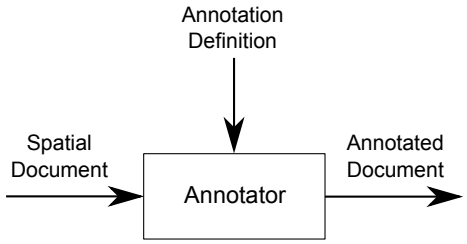
[13] Napríklad text je veľkým tučným písmom

[14] dva odpovedajúce si reťazce musia mať naviac rovnaké veľké a malé písmena

[15] pri porovnávaní na veľkosti znakov nezáleží

[16] často sa nedopatrením užívateľov píšu dvojslovné názvy s veľkými, alebo malými znakmi. V dokumente sa potom objavuje napríklad názov ulice *Veľká Okružná* s veľkým *O*, a v inom dokumente *Veľká okružná* s malým *o*.

[17] príkladom je adresa, ktorá pozostáva z mesta, PSČ a ulice s číslom domu



Obrázek 3.25: Annotator

- hľadaní vzorov v text
- hľadaní už anotovaných fráz v nejakom okolí skúmeného textu
- na postupnom zoskupovaní malých anotovaných fráz do väčších celkov

Napríklad ak je v texte meno ulice a naľavo od neho je číslo, spolu tvoria adresu v ulici. Ak sa pod týmito dvoma frázami naviac nachádza mesto a PSČ, zoskupením týchto fráz je možné vytvoriť konkrétnu adresu.

3.3.2.1 Jednoduchá anotácia

Jednoduchá anotácia je základným prvkom a v podstate je nezávislá na ostatných anotáciách. Musí obsahovať pattern[18], alebo textovú konštantu, typ a pravdepodobnosť, s akou nájdený text odpovedá typu anotácie. Príklad, ako definovať jednoduchý typ je zobrazený v kóde 3.2.

```
1  <simpleType type="city">
2    <const prob="0.5" value="Žilina" matchCase="false"/>
3    <const prob="0.5" value="Bratislava">
4        <context type="zip" prob="0.4" area="row" distance="next" />
5    </const>
6  </simpleType>
```

Kód 3.2: Príkad jednoduchej anotácie

Na riadku 1. je typ anotácie **type** pomenovaný ako *city*. Na riadku 2. a 3. je definované, čo a ako sa ma vyhľadávať a v prípade nájdenia fráze v texte, aká pravdepodobnosť sa má fráze pridadiť. V tomto prípade, ak by anotátor našiel v texte slovo *"Bratislava"*, z odpovedajúceho slova vytvorí frázu typu "city"s pravdepodobnostnou hodnotou 0.5.

Definácia kontextu, ako je na riadku 4., zvyšuje pravdepodobnosť nájdenej fráze. Ak anotátor v jej okolí nájde frázu typu *zip* definovanú v okolí slova *"Bratislava"*, ktorá je najďalej vo vzdialenosti *next* (čo predstavuje susedné slovo/riadok), bola by pravdepodobnosť toho, že slovo "Bratislava"je skutočne typu *city*, zväčšená o 0.4 podľa vzorca 3.1. Teda 0.5+(1-0.5)*0.4 = 0.7 . Viac o atribútoch a hodnotách kontextu je popísané v sekcii Kontext.

Definície typov sa medzi sebou neprepisujú, ale pridávajú. Ekvivalentnou definíciou kódu 3.2 je kód 3.3.

```
1  <simpleType type="city">
2     <const prob="0.5" value="Žilina" matchCase="true"/>
```

[18]Vzor, ktorý definuje množinu reťazcov, ako napríklad regulárny výraz.

```
3  </simpleType>
4  <simpleType type="city">
5     <const prob="0.5" value="Bratislava">
6         <context type="zip" prob="0.4" area="l,d" distance="next" />
7     </const>
8   </simpleType>
```

Kód 3.3: Príkad jednoduchej anotácie

3.3.2.2 Konštanty

Konštanta definuje ako presne vyzerá slovo, ktorému priradí kontext. Príklad definície je uvedený v kóde 3.2. Atribútmi konštanty sú:

- prob - pravdepodobnosť, že nájdené slovo odpovedá typu anotácie. Obsahuje čísla v rozsahu od 0.0 po 1.0
- value - hodnota konštanty
- matchCase - nepovinný atribút - indikuje, či v nájdenom slove musia byť zhodné aj malé a veľké písmená. Môže mať hodnotu *true* a *false*. False je východzia hodnota. V niektorých prípadoch je potrebné, aby záležalo na veľkosti len u prvého znaku. V tom prípade sa nastavuje hodnota *firstletter*.

3.3.2.3 Zoznam z databáze

Pre prehľadnosť konfiguračného súbora a jednoduchú prácu s ním, sa konštanty môžu ukladať aj do databáze[19]. Príklad definície je uvedený v kóde 3.4

```
1 <simpleType type="city">
2   <db table="city_names" where="name␣like␣'A%'" valuecol="name"
3       probCol="prob" matchCase="true"/>
4   <db table="all_cities" probcol="prob" valuecol="name"  />
5 </simpleType>
```

Kód 3.4: Konštanty z databáze

Atribútmi sú:

- table - tabuľka, z ktorej sa budú vyberať konštanty
- where - nepovinný atribút - filtračná podimenka v SQL dotaze. stĺpec
- valuecol - stĺpec, ktorý obsahuje hodnoty konštánt
- probCol - stĺpec, ktorý obsahuje pravdepodobnosť, že nájdené slovo odpovedá typu anotácie. Môže nadobúdať hodnoty od 0.0 po 1.0
- matchCase - nepovinný atribút - indikuje, či v nájdenom slove musia byť zhodné aj malé a veľké písmená. Môže mať hodnotu true a false. False je východzia hodnota.

[19] V konfiguračnom súbore config/config.xml je možné zadať prihlasovacie údaje do databáze.

3.3.2.4 Regulárne výrazy

Ďalším spôsobom, ako určiť, či daná fráza odpovedá nejakému textu je pomocou patternu. Ten v tomto prípade sa zapisuje pomocou regulárneho výrazu. Príklad definície je uvedený v kóde 3.5.

```
<simpleType type="email">
  <regex pattern="[0-9a-z]{3,}@[a-z0-9]+\.[a-z]{2,3}"  prob="0.9" />
  <regex pattern="[0-9a-z]{3,}@[a-z0-9.]+\.[a-z]{2,3}" prob="0.95" /
     ↪ >
</simpleType>
```

Kód 3.5: Regulárny výraz

Atribútmi sú:

- pattern - regulárny výraz
- prob - pravdepodobnosť, že nájdené slovo odpovedá typu anotácie. Obsahuje čísla v rozsahu od 0.0 po 1.0

3.3.2.5 Zmiešaná definícia

Jednotlivé spôsoby definovania anotácie možno kombinovať. Tj. je možne v jednom XML elemente definovať anotačný typ pomocou konštanty, zoznamu z databáze aj pomocou regulárneho výrazu, ako na príklade 3.6.

```
<simpleType type="phone">
  <const prob="0.8" value="112" />
  <const prob="0.8" value="150" />
  <const prob="0.8" value="155" />
  <const prob="0.8" value="158" />
  <const prob="0.8" value="911" />
  <db table="commercial_numers" probcol="prob" valuecol="number"  />
  <regex pattern="\+420([␣]*[0-9]){7,10}" prob="0.99" />
</simpleType>
```

Kód 3.6: Zmiešaná definícia

3.3.2.6 Kontext

Kontext, v ktorom sa vyšetrovaná fráza nachádza, upresní jej pravdepodobnostné ohodnotenie. Príkladom je Žilina. Ak by sa v okolí nachádzalo krstné meno, bude Žilina s veľkou pravdepodobnosťou priezvisko. Ak by sa v okolí nachádzalo PSČ, bude Žilina v veľkou pravdepodobnosťou mestom. Pre definíciu kontextu je potrebné definovať polohu, vzdialenosť a typ fráze (túto frázu budeme označovať ako *kontextová fráza*), ktorá tvorí kontext.

```
<simpleType type="city">
  <db table="cities" probcol="prob" valuecol="name"  />
    <context type="PSC" prob="0.9" area="ROW,U,D"
      distance="close" minprob="0.5" />
  </db>
  <const prob="0.8" value="Žilina">
    <context .../>
  </const>
  <regex pattern="\[A-Z][a-z]{2,20}" prob="0.1" />
```

```
10      <context .../>
11    </regex>
12  </simpleType>
```

Kód 3.7: Príklad definície kontextu

Kontext má tieto atribúty:

- *type* - typ kontextovej fráze, ktorá sa bude hľadať v okolí
- *prob* - kontextová pravdepodobnosť, ktorá ovplyvňuje pravdepodobnosť aktuálne vyšetrovanej fráze. Podrobne je popísaná v sekcii Pravdepodobnosť.
- *area* (nepovinný atribút) - oblasť, v ktorej sa kontextová fráza nachádza. Podrobne je popísaná v sekcii Oblasť. Východzia hodnota je *"all"*.
- *distance* (nepovinný atribút) - vzdialenosť, v akej sa kontextová fráza nachádza. Podrobne je popísaná v sekcii Vzdialenosť. Východzia hodnota je *"ignore"*
- *minprob* (nepovinný atribút) - ak je pravdepodobnosť kontextovej fráze menšia, ako minprob, bude sa ignorovaná.

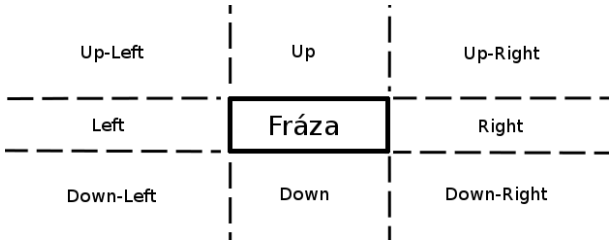
3.3.2.7 Pravdepodobnosť

Kontext v prípade pozitívneho nálezu pridáva pravdepodobnosť vyšetrovanej fráze. Hodnotu, ktorá ovplyvňuje pravdepodobnosť vyšetrovanej fráze, budeme označovať ako *Kontextovú pravdepodobnosť*. Tá môže nadobúdať hodnoty od 0.0 po 1.0. Predpokladajme, že aktuálna pravdepodobnosť vyšetrovanej fráze je p_x. Ak je hodnota kontextovej pravdepodobnosti p_k, potom výsledná pravdepodobnosť $\bar{p}_x$ bude vypočítaná poľa rovnice 3.1

$$\bar{p}_x = p_x + (1 - p_x) * p_k \tag{3.1}$$

3.3.2.8 Oblasť

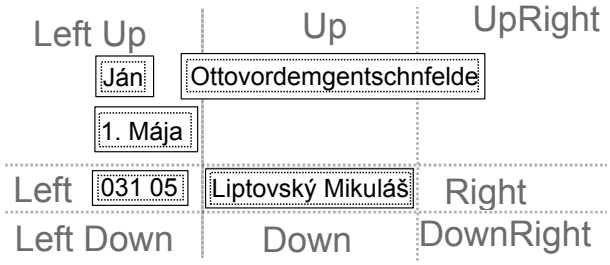
Každá fráza rozdeľuje svoje okolie na 8 oblastí podľa smeru: hore, dole, vľavo, vpravo a ich kombinácie, ako je zobrazené na obrázku 3.26.



Obrázek 3.26: Relatívne oblasti

Pri definovaní oblasti bude anotátor klásť dôraz tiež na to v akom smere kontextová fráza nachádzda. Ak si zabalíme každú frázu do obdĺžnika, ako na obrázku 3.27, zistíme, že fráze môžu zasahovať do dvoch, alebo aj troch oblastí naraz. Pre vyšetrovanú frázu *Liptovský Mikuláš* by jej kontextová fráza *Ottovordemgentschnfelde* nachádzala v jej Up-Left, Up a Up-Right oblasti. Naopak pre

vyšetrovnú frázu *Ottovordemgentschnfelde* by sa jej kontextová fráza *Liptovský Mikuláš* nachádzdala len v jej Down oblasti.



Obrázek 3.27: Príklad vzájomnej polohy

Pri definícií, sa pre označovanie oblastí používajú skratky:

- *u* - Up oblasť
- *ul* - Up-Left oblasť
- *ur* - Up-Right oblasť
- *r* - Right oblasť
- *l* - Left oblasť
- *d* - Down oblasť
- *dl* - Down-Left oblasť
- *dr* - Down-Right oblasť
- *row* - Left a Right oblasť
- *col* - Up a Down oblasť
- *all* - všetky oblasti

Pri definícií, je možné oblasti kombinovať. Skratky *ROW* a *COL* len nahradzujú kombináciu *L,R* a *U,D*.

3.3.2.9 Vzdialenosť

Okrem smeru, respektíve oblasti, kde sa kontextová fráza nachádza, je často dôležitá aj vzdialenosť:

- *next* - kontextová fráza sa nachádza v bezprostrednej blízkosti vyšetrovanej fráze (tj. do vzdilenosti 1. slova).
- *near* - kontextová fráza sa nachádza maximálne do vzdialenosti 3 slov/riadkov od vyšetrovanej fráze.
- *close* - kontextová fráza sa nachádza maximálne do vzdialenosti 5 slov/riadkov od vyšetrovanej fráze.
- *ignore* - východzia hodnota. Udáva, že na vzdialenosti nezáleží.

3.3.2.10 Komplexná anotácia

Komplexná anotácia má predovšetkým zoskupovať nájdené anotované fráze. Principiálne *Komplexná anotácia* vychádza z už nájdenej anotovanej fráze. V jej okolí hľadá anotácie, ktoré do seba *komplexná anotácia* zahrnie. Rovnako ako *jednoduché anotácie*, používa *komplexná anotácia* pre zväčšenie svojej pravdepodobnosti *kontextové anotácie*. Naviac dokáže vo svojom okolí anotovať text. Napríklad ak *komplexnou anotáciou* je adresa, potom riadok nad adresou anotuje ako meno prijímateľa.

```
<complexType type="address">
  <phrase from="street" minprob="0.4">
    <contains type="city" minprob="0.4" area="d" distance="close" />
    <contains type="PSC" minprob="0.4" area="row" distance="close"
        relativeTo="city" />
    <define type="contactName" direction="U" size="phrase"/>
  </phrase>
  <phrase from="streetAddr" minProb="0.4" incProb="0.8">
    <contains type="cityzip" minProb="0.4" area="d" distance="next"
       ↪  />
    <contains type="companyForm" minProb="0.4" area="u" distance="
       ↪ near"
        relativeTo="all"/>
    <define type="contactName" direction="l" size="phrase" include="
       ↪ true"
        relativeTo="companyForm"/>
  </phrase>
  ...
</complexType>
```

Kód 3.8: Príklad definície komplexného typu

Príklad definície je zobrazený v kóde 3.8. Komplexná anotácia sa definuje elementom *complexType* s pomenovaním typu, ktorý sa udáva do atributu *type*. Pravidlá pre tvorbu komplexnej anotácie sa zadávajú do elementu *phrase*. Do neho sa zadávajú dva atibúty:

- *from* - typ východzej anotovanej fráze
- *minprob* (nepovinný atribút) - minimálne pravdepodobnosť východzej anotovanej fráze
- *incProb* (nepovinný atribút) - výsledná pravdepodobnosť sa počíta ako priemer nájdených fráz (tj. východzej fráze a fráz z *"contains"*). Táto hodnota výslednú pravdepodobnosť zväčší podľa vzorca 3.1.

Podelementom *"contains"* sa definuje, aké ďalšie anotované fráze má do seba komplexná anotácia zahrnúť. Definícia je podobná ako definícia kontextu (viď. sekcia Kontext). Chýba jej atribút *prob*, zato naviac je možné definovať referenčnú frázu. To jest frázu, ktorá slúži ako východzí bod pre prehľadávanie okolia.

Element "contains"má tieto atribúty:

- *type* - typ fráze, ktorá sa bude hľadať v okolí
- *minprob* (nepovinný atribút) - nájdená frázam musí mať väčšiu alebo rovnakú pravdepodobnosť, ako minprob

- *area* (nepovinný atribút) - oblasť, v ktorej sa fráza nachádza. Podrobne je popísaná v sekcii Oblasť. Východzia hodnota je *"all"*.

- *distance* (nepovinný atribút) - vzdialenosť, v akej sa fráza má nachádzať. Podrobne je popísaná v sekcii Vzdialenosť. Východzia hodnota je *"ignore"*

- *relativeTo* (nepovinný atribút) - ktorá fráza z už zahrnutých má slúžiť ako východzí bod pre prehľadávanie. Je tiež možné zadať hodnotu *all*. Z doposiaľ zahrnutých fráz sa vytvorí jedna dočasná a ta slúži ako vychodzia fráza pre prehľadávanie.

V prípade, že je komplexná fráza nájdená, je možné definovať v okolí novú frázu. Napríklad v prípade nájdenia adresy je možné definovať meno prijímateľa, ktoré môže byť ťažké definovať (napríklad ak je to spoločnosť, alebo inštitúcia).

- *type* - nájdená fráza bude anotovaná týmto typom

- *direction* - smer, v ktorej sa fráza nachádza. Hodnoty smeru sú rovnaké, ako *area*, ktorá je popísana v sekcii Oblasť. Ale *direction* môže označovať len jednu oblasť. Teda len *up*, *up-left*, *up-right* a tak podobne, ale nie *col* alebo *row*.

- *size* - akú veľkú frázu má anotátor anotovať. Môže nadobúdať hodnoty:

  - word - vyhľadá a anotuje len najbližšie *SpatialWord* v danom smere
  - phrase- su to všetky slová v najbližšej *SpatialLine*, zoradené od najbližšieho, až po slovo s oddeľovacím znakom. Oddeľovací znak sa definuje v *anotačnom súbore* a sú to štandardne znaky ";,:". Hlavným cieľom je nájsť v riadku frázu, ktorá sa nachádza medzi čiarkami, ako napríklad: "Liptovský Mikuláš, 031 05, Slovensko"
  - line - vyhľadá a anotuje len najbližšiu *SpatialLine* v danom smere

- *relativeTo* (nepovinný atribút) - od ktorej fráze sa má výľadávať. Tiež môže nadobôdať hodnotu *all*, čiže dočasne sa vytvorí fráza z doposiaľ nájdených fráz a od nej sa bude vyhľadávať. Východzia hodnota je *"all"*

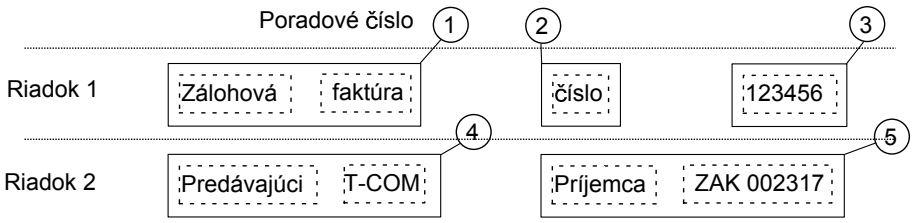
- *include* (nepovinný atribút) - môže nadobúdať hodnoty *"true"* alebo *"fasle"*. Určuje, či novo definovaná fráza sa má zahrnúť do súčasnej komplexnej anotácie. Východzia hodnota je *"true"*.

3.3.3 Vyhľadávanie a anotovanie fráz

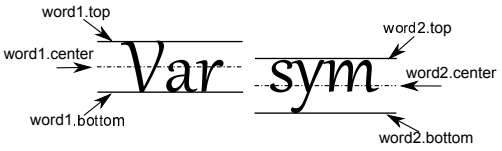
3.3.3.1 Usporiadanie a očíslovanie SpatialLine

Aby bolo možné priestorovo vyhľadávať text v dokumente, je nutné očíslovať *SpatialLine*. Jednak je potrebné priradiť im číslo riadku v dokumente a poradové číslo. Číslom riadku v dokumente je myslené, na koľkom riadku sa *SpatialLine* nachádza (viď obrázok 3.28).

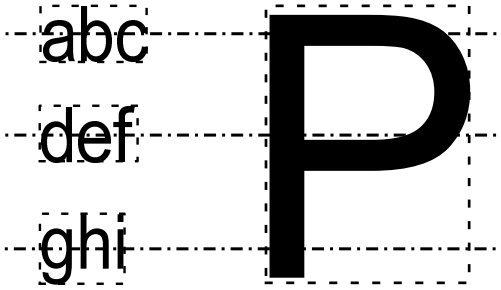
Pri prevode textového súboru do *SpatialDocument*u je určenie čísla riadku jednoduché, nakoľko sú všetky riadky a slová rovnako vysoké a Y-ové súradnice sú v každom riadku rovnaké. Ak však bol *SpatialDocument* vytvorený z obrázku,



Obrázek 3.28: Očíslovanie *SpatialLine*



Obrázek 3.29: Slová rovnakého riadku



Obrázek 3.30: Slová rôznych riadkov

tak nielenže v jednom riadku sú slová, ktoré majú rôznu výšku, ale dokonca hrozí že posledné slovo v riadku môže pri miernom sklone byť o "riadok vyššie"než slovo na začiatku dokumentu[20]. Pre určenie, či *SpatialWord* alebo *SpatialLine* sa nachádzajú na tom istom riadku, bolo vytvorené pravidlo:

Ak vertikálny stred jedného SpatialWord sa nachádza medzi hornou a dolnou hranicou druhého SpatialWord a takiež ak vertikálny stred druhého SpatialWord sa nachádza medzi hornou a dolnou hranicou prvého SpatialWord, potom obe SpatialWord sú na rovnakom riadku.

Toto pravidlo sa označuje ako **Pravidlo rovnakého riadku**. Ak by sme prvé slovo označili ako word1 a druhé ako word2, potom musí platiť:

$$word1.top \leq word2.verticalCenter \leq word1.bottom$$

a

$$word2.top \leq word1.verticalCenter \leq word2.bottom$$

Obrázok 3.29 ilustruje pravidlo, ktoré rozhoduje, či dve slová sú na tom istom riadku. Je na ňom vidieť, že vertikálne stredy sa nachádzajú medzi hornou a dolnou hranicou druhého slova. Obrázok 3.30 zas ilustruje extrémnejší prípad. Z *pravidla rovnakého riadku* vyplýva, že len slovo *def* a písmeno *P* sú na rovankom riadku. Slová *abc* a *ghi* s písmenom *P* splňujú len prvú polovicu pravidla, ale nie druhú(to jest:*"stred druhého SpatialWord sa nachádza medzi hornou a dolnou hranicou prvého"*).

V očíslovaných riadkoch je teraz jednoduché prehľadávať riadok "nad", alebo "pod", alebo prehľadávať susedné *SpatialLine*.

[20]ak na papieri formátu A4, je vytlačený text s veľkosťou písma 10pt, stačí, aby bol dokument pootočený o 1 stupeň a slovo na konci a začiatku riadku sa po Y-ose vôbec neprekrývajú a algoritmus vyvodzuje, že sú na rozdielnych riadkoch

Kebyže na vyhľadávanie použijeme R-strom, bolo by oveľa ťažšie a časovo náročnejšie takéto slová nájsť. Napríklad ak by sme chceli nájsť slová z predchádzajúceho riadku, museli by sme určiť oblasť z ktorej chceme vyhľadať slová podľa veľkosti písma, medzery medzi riadkami a predpokladať, že písmo je rovnako veľké. Problémom sú prípady, keď najbližší riadok je takzvane ob-jedno[21]. Iným spôsobom využitia R-stromov by bolo vybrať slová z väčšej oblasti, a zoradiť ich podľa vzdialenosti. Tu je problém z blízkymi slovami, a určením, ktoré slová sú skutočne z predchádzajúceho riadku.

Po očíslovaní *SpatialLine* sa všetky zoradia podľa poradového čísla. V každom *SpatialLine* naviac treba zabezpečiť, že *SpatialWord* budú zoradené podľa x-ovej súradnice. Teraz je možne každému *SpatialWord*u priradiť nasledujúce a predchádzajúce *SpatialWord*. Tieto ukazovatele sa používajú v algoritmoch, ktoré potrebujú prehľadávať len susedné slová.

3.3.3.2 Predfiltrovanie slov

SpatialDocument ktorý bol vytvorený pomocou OCR modulu, často obsahuje reťazce, ktoré nie sú slovami. OCR aplikácia sa snaží rozoznať slovo z čiar kriviek, ktoré sú obrázkami, logom, pečiatkou, alebo je špina na papieri, pokrčenie oskenovaného dokumentu[22] alebo dokonca kancelárska spinka ponechaná na papieri. Výsledkom je napríklad reťazec "—__.;!___—". Slová, ktoré neobsahujú abecedný, alebo číselný znak sa do výsledného dokumentu nepridávajú.

3.3.3.3 Inicializácia anotácií

Trieda *AnnotateBuilder* z *Anotačného súboru config/annotate.xml* načíta všetky anotačné pravidlá do inštancie triedy *AnnotationDefinition*. Tá funguje ako datakeeper[23] anotačných pravidiel. *AnnotateBuilder* následne vytvára inštanciu triedy *Annotator*, čo je hlavná trieda, ktorá anotuje text. *AnnotateBuilder* postupne z *AnnotationDefinition* vyberá definované pravidlá, generuje anotácie[24] a pridáva ich do *Annotator*u.

3.3.3.4 Vyhľadávanie konštánt

Konštanty sa v *Anotačnom súbore* definujú buď ako:

- *const* - vytvorí sa jedna konštanta
- *db* - z databáze sa z definovaného stĺpca vyberú riadky a z nich sa vytvoria konštanty

Pri rozhodovaní, akú štruktúru použiť pre čo najrýchlejšie vyhľadávanie konštánt v texte, respektíve v *SpatialDocument*e, vyšla ako najneefektívnejšia štruktúra

[21] často sa uvádza meno prijímateľa, prázdny riadok, adresa prijímateľa

[22] pokrčenie spôsobuje tieň, ktorý je tmavší než papier a OCR si teda myslí, že to môže byť znak

[23] Datakeeper je trieda, ktorej hlavným cieľom je udržiavať dáta a zjednodušovať prístup k nim

[24] anotácie sú inštancie tried, pomocou ktorých *Annotator* anotuje text

hešovacia tabuľka, respektíve hešovaia mapa[25]. Vyplývalo to z výsledkov uvedených v sekcii 5.1 Vyhľadávanie slov v texte.

Annotator preto pre vyhľadávanie konštánt v *SpatialWord*e používa hešovaciu mapu. Hešovacia mapa používa ako kľúč objekty typu String a ako hodnoty používa pole anotácií. Napríklad kľúč *Žilina*"bude odkazovať na pole, ktoré obsahuje anotácie *[(mesto,40%),(priezvisko,20%)]*. Teda ak hešovacia mapa obsahuje slovo z *SpatialWord*u, priradí mu zoznam odpovedajúcich anotácií.

Annotator prehľadáva *SpatialDocument* po slovách: z každého *SpatialLine* sa vyberú slová a tie sa vyhľadávajú v hešovacej mape. Nájdenie odpovedajúcich anotácií je jedno-slovné konštanty je teda jasné.

Pre vyhľadávanie konštanty, ktorá pozostáva z dvoch a viacerých slov, napríklad "Liptovský Mikuláš", je potrebné rozšíriť spôsob vyhľadávania. Konštanta sa rozdelí na slová. Prvé slovo sa použije ako vyhľadávací kľúč do hešovacej mapy a ostatné slová sa uložia do anotácie. Pri vyhľadávaní sa potom dodatočne overuje každá anotácia, či je konštanta anotácie jedno, alebo viacslovná. Ak je viacslovná, potom sa musia prehľadať aj susedné slová vyšetrovaného *SpatialWord*. Vďaka tomu, že každé *SpatialWord* má ukazateľ na následujúce slovo (viď sekcia Usporiadanie a očíslovanie SpatialLine) stačí vziať odpovedajúci počet susedných slov a tie porovnať so slovami, ktoré sú uložené v anotácii (jej konštante).

V definícií anotácie konštanty je možné definovať, či pri porovnávaní záleží na veľkosti znakov. *Annotator* v skutočnosti obsahuje tri hešovacie mapy, kde každá odpovedá jednému zo spôsobov porovnávaní:

- *Match case* - záleží na veľkosti znakov
- *Ignore case* - nezáleží na veľkosti znakov
- *First letter* - len pri prvom znaku záleží na veľkosti

Porovnávanie typu *Match case* je popísané v predchádzajúcom odstavci.

Pri porovnávaní typu *Ignore case* je konštanta v anotácilí prevedená na malé písmená a následne uložená do hešovacej mapy. Táto hešovacia mapa je označená ako *constantsIgnoreCase*[26]. Pri vyhľadávaní je slovo zo *SpatialWord*u tiež prevedené na malé písmená a následne hľadané v hešovacej mape *constantsIgnoreCase*.

Pri porovnávaní typu *First letter* je to podobné, ale len u prvého znaku je zachovaná veľkosť a hešovacia mapa je označená ako *constantsFirstLetterMatch*.

3.3.3.5 Vyhľadávanie regulárnych výrazov

Pri vyhľadávaní podľa regulárnych výrazov sa všetky *SpatialWord*y prevedú do jedného reťazca, pričom je zachované ich poradie. Pri prevode sa tiež zaznačujú poradové čísla *SpatialWord* slov v reťazci, aby sa dali spätne dohľadať. Obrázok 3.3.3.5 ilustruje reťazec s uloženými poradovými číslami *SpatialWord*[27].

Annotator potom postupne prechádza všetky anotácie s regulárnymi výrazmi a v reťazci hľadá ich výskyty. V prípade nálezu dohľadá podľa pozície *SpatialWord*y a priradí im anotáciu.

[25]HashMap je implementácia hešovacej tabulky v jazyku Java
[26]V triede *Annotator* má táto hešovacia mapa názov *constantsIgnoreCase*
[27]v skutočnosti sa ukladajú do intervalového poľa kôli úspore pamäte

		L	i	p	t	o	v	s	k	ý		M	i	k	u	l	á	š	
		1	1	1	1	1	1	1	1	1		2	2	2	2	2	2	2	

Obrázek 3.31: Indexácia *PhraseWords* v reťazci

3.3.3.6 Nájdenie kontextu

Annotator v prvom kroku hľadá v dokumente jednoduché anotácie. Anotované fráze ukladá priamo do *spatialdocument*u.Tie potom prechádza a hľadá, či majú definovaný aj kontext. V prípade, že áno, začne v okolí vyšetrovanej fráze hľadať frázu odpovedajúcu kontextu. Fráza, ku ktorej *Annotator* hľadá kontext sa označuje ako **vyšetrovaná fráza**. Príkladom môže byť vyšetrovaná fáza *Liptovský Mikuláš* a v *anotačnom súbore* je definované pravidlo uvedené na obrázku 3.9. Podľa tohto príkladu by *Annotator* na naľavo, napravo a pod frázou *Liptovský Mikuláš* hľadal frázu anotovanú ako *zip* (teda PSČ).

```
...
<simpleType type="city">
  <db table="cities" probcol="prob" valuecol="name"  />
    <context type="zip" prob="0.9" area="L,R,D"
       distance="next" minprob="0.5" />
  </db>
</simpleType>
...
```

Kód 3.9: Príklad kontextu

Prehľadávanie susedných fráz prebieha v dvoch krokoch. Najprv sa prehľadávajú fráze smerom k začiatku dokumentu a v druhom kroku smerom ku koncu dokumentu. Maximálna vzdialenosť, do ktorej sa prehľadáva je daná hodnotou *distance* v definícií pravidla. Hodnota *distance* je popísaná v sekcii Vzdialenosť. Tá môže určovať, že maximálna vzdialenosť kontextovej fráze je 1,3,5 alebo na vzdialenosti nezáleží (tá je reprezenzovaná hodnotou 10 000, pretože sa predpokladá, že dokument nebude mať viac ako 10 000 riadkov). Odpočítaním/pripočítaním tejto hodnoty k číslu riadku *vyšetrovanej fráze* je definovahá hodnota **minRowNumber** a **maxRowNumber**.

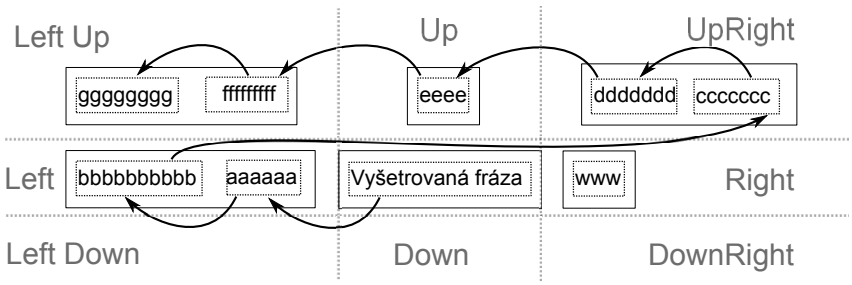
Pseudo kód algoritmu prehľadávania je založený na syntaxii jazka Java. Premenná oblasť je bitová maska, ktorá z refinície pravidla kontextu určuje oblasti, v ktorých sa má kontextová fráza vyhľadávať, preto operátor {& je braný ako bitový operátor a operátor *&&* je braný ako logický AND. Je tiež dôležité si uvedomiť, že vyšetrovaná fráza môže byť na niekoľkých riadkoch. Pseudokód algoritmu je uvedený v kóde 3.10.

```
line = actual_phrase.previous
while(line.line_number >= minRowNumber && line.line_number>=0) {
   rozdel slova riadku line do: laveSlova, stlpcoveSlova,
      ↪ praveSlova
   if (riadok >= horny riadok vysetrovanej fraze)
      //prehladavanie na urovni vysterovanej fraze
      if (oblast & left > 0)  najdi frazu v laveSlova
      if (oblast & right > 0) najdi frazu v praveSlova
   else
```

```
9         //prehladavanie nad urovnou vysetrovanej fraze
10        if (oblast & leftUp > 0 ) najdi frazu v laveSlova
11        if (oblast & up > 0 )     najdi frazu v stlpcoveSlova
12        if (oblast & rightUP > 0) najdi frazu v praveSlova
13  }
```

Kód 3.10: Algoritmus vyhľadávania kontextových fráz

Na obrázku 3.32 je ilustorvaná poradie, v akom sa kontextové fráze vyhľadávaju oblasti, do ktorých slová spadajú.



Obrázek 3.32: Poradie prehľadávania kontextových fráz

Prehľadávanie smerom ku koncu dokumentu sa vykonáva obdobne.

V prípade nájdenia kontextovej fráze sa overí, či spĺňa aj podmienku minimálnej pravdepodobnosti (tá je definovaná parametrom *minprob*). Podľa vzorca pre výpočet pravdepodobnosti, uvedenom v sekcii Pravdepodobnosť, sa pre aktuálne nájdenú kontextovú frázu vypočíta nová pravdepodobnosť *vyšetrovanej fráze*.

Ak má jedna fráza v anotačnom pravidle definovaných viac kontextových pravidiel, potom vyhľadávanie a výpočet výslednej pravdepodobnosti *vyšetrovanej fráze* sa robí nezávisle od ostatných kontextových pravidiel. Teda ak prvá kontextová fráza zvýšila výslednú pravdepodobnosť napríklad z 0.4 na 0.7, potom druhá kontextová fráza zvýši pravdepodobnosť z 0.7 napríklad na 0,85 (nie z 0.5 na 0.75). Z toho jasne vyplýva, čim viac kontextových pravidiel má vyšetrovaná fráza, tým vyššiu pravdepodobnosť môže mať.

3.3.3.7 Vyhľadávanie komplexných anotácií

Komplexné anotácie na rozdiel od jednoduchých nemajú definované vzory (patterny), ktoré by vyhľadávali v texte dokumentu. Komplexné anotácie sú závislé na nájdených jednoduchých anotáciách. Ich zjedodušený princíp je následovný:

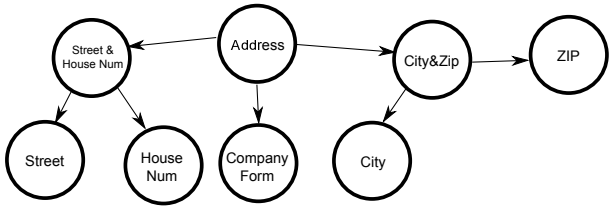
1. nájdi frázu konkrétneho typu
2. v jej okolí nájdi ďalšie anotované fráze (podobne ako kontext) a zahrň ich do tejto anotácie
3. (ak je definované) nájdi kontext v okolí a zvýš pravdepodobnosť tejto fráze[28]
4. (ak je definované) v určenom smere vyber slovo/riadok a ten definuj ako novú frázu

[28] rovnako ako kontext funguje o jednoduchej anotácie

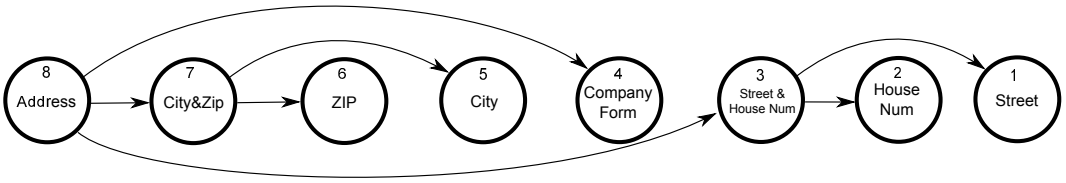
Príprava

Z definície *komplexných anotácií* (viď sekcia Komplexná anotácia) plynie závislosť (z definície *"from"* a definície *"contains"*), kde jedna komplexná anotácia môže je závislá buď na *jedoduchej anotácií*, alebo na inej *komplexnej anotácií*. Pri v momente hľadania komplexnej je teda potrebné mať už nájdené anotácie, na ktorých aktuálne vyšetrovaná anotácia zavisí. Preto je portebné určiť poradie, v akom sa budú komplexné anotácie v dokumente vyhľadávať. Problémom by bolo, ak by komplexné anotácie boli závislé cyklicky. V tom prípade by sa nedalo určiť poradie, v akom by sa mali komplexné anotácie vyhľadávať. Preto cyklické závislosti budú zakázané.

Pre vyriešenie usporiadania, je potrebné previesť závislosti komplexných anotácií do grafu. Obrázok 3.33 ilustruje príklad, ako by taký graf vyzeral. Graf pomocu topologického usporiadania[8] zoradíme tak, aby všetky závislosti (šípky) vpravo. Tie najkomplexnejšie anotácie sa budú tým pádom nachádzať vľavo a tie nezávislé v pravo. Pri usporiadavaní sa zároveň zistí, či graf obsahuje, alebo neobsahuje cyklus. V prípade, ak by sa cyklus našiel, aplikácia zobrazí upozornenie o chybnej konfigurácií *anotačného súboru* a skončí. Usporiadaný graf je zobrazený na obrázku 3.34.



Obrázek 3.33: Príklad neusporianých závislostí prevedených do grafu



Obrázek 3.34: Príklad usporiadaných závislostí prevedených do grafu

Hľadanie

Výsledný zoznam je prechádzaný od zadu (teda od najjednoduchšieho typu), pričom sa vyhľadávajú len komplexné anotácie (jednoduché anotácie boli už nájdené v predchádzajúcich krokoch, viď kapitoly Vyhľadávanie konštánt a **??**). V algoritme je použitá funkcia *"nájdi anotáciu typu X v okolí"*. Pre nájdenie anotácie sa používa algoritmus pre nájdenie kontextu v okolí, popísaný v kóde 3.10 (viď sekcia Nájdenie kontextu).

Pseudokód algoritmu je uvedený v kóde 3.11. Ten hľadá frázu v dokumente pre konkrétnu komplexnú anotáciu. Vhľadávanie sa spúšťa zavolaním funkcie *findPhraseForAnnoation()* (pseudokód vychádza z jazyku Java) :

```
1
2  function findPhraseForAnnoation(komplexAnotacia){
3    F = komplexAnotacia.from //typ vzchodzej anotovanej fraze
4    najdeneKomplexneFraze= new Array()
5    for each( basePhrase : najdi v dokumente fraze typu F){
6        najdeneFraze=getRelativePhrases(komplexAnotacia,
7                        [basePhrase],
8                        komplexAnotacia.contains)
9        najdeneKomplexneFraze.append(najdeneFraye)
10   }
11   for each( foundPhrase: najdeneKomplexneFraze){
12      najdi a vytvor frazu komplexAnotacia.define v okoli foundPhrase
13   }
14 }
15
16 function getRelativePhrases(komplexAnotacia,
17                             foundPhrases,
18                             phrasesToSearch){
19   //ak netreba vyhadavat dalsiu anotovanu frazu,
20   if (phrasesToSearch.isEmpty==true){
21     // vytvor novu komplexnu anotovanu frazu a vrat ju v poli
22     return [new ComplexPhrase(foundPhrases,komplexAnotacia)]
23   }
24
25   if (phrasesToSearch[0].relativeTo == 'all'){
26     //vytvorenie docasenej fraze
27     relativnaFraza = new ComplexPhrase(foundPhrases,komplexAnotacia)
28   } else {
29     relativnaFraza = najdi v foundPhrases frazu
30                      typu phrasesToSearch[0].relativeTo
31   }
32
33   X = phrasesToSearch[0].type
34   trebaNajstFraze= odober 0-ty prvok pola phrasesToSearch
35
36   najdeneKomplexneFraze = new array
37   for each(najdenaFraza: najdi anotacie typu X v okoli
38                                  fraze relativnaFraza){
39     doposialNajdeneFraze= foundPhrases.clone()
40     doposialNajdeneFraze.append(najdenaFraza)
41     // rekurzivne
42     najdeneFraze= getRelativePhrases(komplexAnotacia,
43                                      doposialNajdeneFraze,
44                                      trebaNajstFraze);
45
46     najdeneVyslendeKomplexneFraze.append(najdeneFraze)
47   }
48   return najdeneKomplexneFraze
49 }
```

Kód 3.11: Algoritmus pre vyhľadávanie komplexných fráz

Na začiatku sa v dokumente vyhľadajú všetky anotované fráze typu, ktorý je definovaný v komplexnej anotácií v atribúte *"from"*. Pre nájdenú frázu sa

následne zavolá funkcia *"getRelativePhrases()"*, ktorá vráti pole nájdených komplexne anotovaných fráz.

Ako je v kóde 3.11 vidieť, vyhľadávanie *"contains"* fráz (teda ďalších fráz, ktoré tvoria komplexne anotovanú frázu) je použitá rekurzívna funkcia. Pretože prehľadávanie fráz definovaných v *"contains"* je závislé od doposiaľ nájdených fráz a oblasť, v ktorej sa bude fáza vyhľadávať (definovaná parametrom *"relativeTo"*) závisí práve na doposiaľ nájdených fráz, vedie spôsob hľadania k prehľadávaniu typu *"do hĺbky"*. To z dôvodu, že v každom bode je nutné si udržiavať informáciu, "odkiaľ sme prišli", čo v tomto prípade je pole fráz, ktoré sa doposiaľ našli.

Rekurzívna funkcia *"getRelativePhrases()"* najprv skontroluje pole *"phrasesToSearch"*, teda aké ďalšie anotácie treba hľadať:

- Ak je pole prázdne, je jasné, už žiadnu frázu netreba hľadať a teda stačí vytvoriť komplexne anotovanú frázu a tú vrátiť. Pretože ale celkovým výsledkom funkcie pole anotovaných fráz, je nutné túto frázu zabaliť do pola.
- Ak ale pole prázdne nie je:
  - vyberie z poľa *"phrasesToSearch"* prvú definíciu (ktorá sa definuje v *anotačnom súbore* ako element *"contains"*) a nájde podľa tejto definície všetky anotované fráze v dokumente (anotované fráze teda musia odpovedá oblasti, v ktorej sa majú nachádzať, musia mať odpovedajúci typ a tiež pravdepodobnosť)
  - potom pre každú nájdenú frázu : vytvorí nove pole z doposiaľ nájdených fráz a prídá k nim nájdenú frázu, z poľa *"phrasesToSearch"* odoberie prvý prvok a rekurzívne sa zavolá funkcia *"phrasesToSearch"*.
  - všetky výsledky z rekurzívneho volania sa uložia to poľa v tie funkcia vráti ako výsledok

Pretože s každým volaním je odobraný jeden prvok z poľa *"phrasesToSearch"*, je rekurzívne volanie deterministické.

Po vyhľadaní

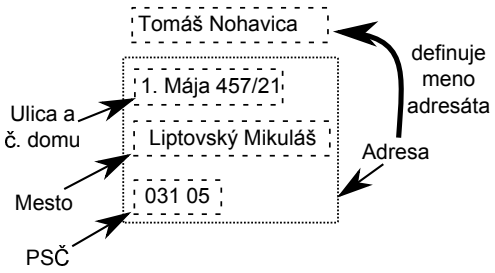
Potom, ako je komplexná fráza nájdená, *Annotator* skontroluje či definícia komplexnej anotácie obsahuje element *"define"*. Ten totiž umožňuje anotovať frázu, ktorá sa nachádza okolí komplexne anotovanej fráze.

Na obrázku 3.35 je príklad, ako komplexne anotovaná fráza môže anotovať text vo svojom okolí. Nájdená komplexne anotovanán fráza je *adresa*.Tá pozostáva z ulice, čísla domu, mesta a PSČ. Ak je definícia "define"rovnaká, ako v príklade uvedom v kóde 3.12, potom *Annotator* nad komplexne anotovanou frázou (pretože parameter *"direction"* je nastavená na *"u"*, čo je *"UP"* a *relativeTo="all"*, čo označuje celú komplexnú frázu) vyberie celý riadok (kvôli parametru *size="line"*). Z tohoto riadku vytvori novú frázu s typu *"contactName"* a začlení ho do seba (kvôli parametru *include="true"*).

Grafické znázornenie príkladu je na obrázku 3.35. Je tam vidieť, že komplexná fráza typu "adresa"definuje z riadku nad ňou novú frázu.

```
<complexType type="address">
  <phrase from="streetAddr" minProb="0.4" incProb="0.8">
     ...
   <define type="contactName" direction="u" size="line" include="
      ↪ true"
        relativeTo="all"/>
  </phrase>
  ...
</complexType>
```

Kód 3.12: Príklad definície "define"komplexného typu



Obrázek 3.35: Príklad ako komplexná anotácia identifikuje novú frázu

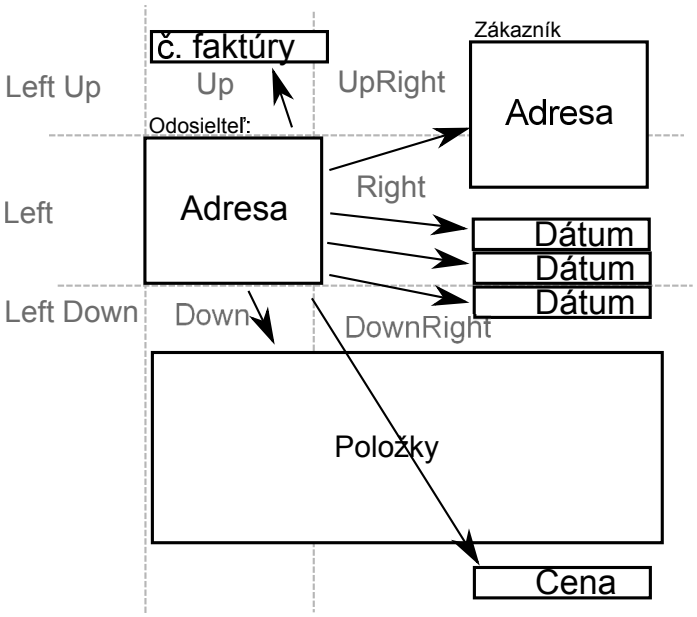
3.4 Vyhľadávač dokumentov

Celé vyhladávanie podobných dokumentov sa vykonáva v daztabázi. Preto je nutné celý dokument spolu so všetkými frázami uložiť do databáze. Pre vyhľadávanie podovných dokumentov sa z nájdených fráz použijú len tie, ktoré majú pravdepodobnosť väčšiu ako 90%. Táto hodnota vychádza z nastavenia *konfiguračného súboru*, kde fráza s hodnou väčšou ako 0.9 skutočne odpovedá svojmu typu. V prípade, že fráza je súčasťou komplexnej fráze, vyberie sa len rodičovská komplexná fráza. Takéto fráze sa označujú ako *ImportantPhrase*. V bežnom dokumente ich býva do 10-tich. Vzájomná poloha a typ *ImportantPhrase* fráz bude tvoriť vyhľadávací kľúč pre daný dokument.

ImportantPhrase budú rovnaké.

Hlavou myšlienkou pri vyhľadávaní podobných dokumentov je nájsť dokument, ktorý má *ImportantPhrase* rovnako rozložené, teda že vzájomné polohy *ImportantPhrase* sú rovnaké. Vzájoumnou polohou sa rozume oblasť, do ktorej spadá jedna *ImportantPhrase* voči druhej. Na obrázku 3.36 do akých oblastí spadajú *ImportantPhrase* voči fráze *adrese odisielateľa*. Ako je vidieť, tak:

- *čislo faktúry* spadá do oblasti "Up"a "UpRight",
- *adresa zákazníka* spadá do oblasti "UpRight"a "Right",
- *dátumy* spadajú do oblasti "Right"jeden do oblasti "RightDown",
- *položky* spadá do oblasti "Down"a "DownRight",
- *cena* spadá do oblasti"DownRight",



Obrázek 3.36: Vzájomné polohy *ImporetantPhrases*

Každá oblasť má priradené svoje číslo, ktoré je mocninou čísla dva (tj 1,2,4,8...). To umožňuje ukladať vzájomnú polohu ako bitovú mapu. Vďaka tomu sa dá priamo v databáze pomocou bitových operátorov porovnať či dve podobné fráze z dvoch rôznych dokumentov majú na rovnakých miestach rovnaká typy fráz.

ImportantPhrase je treba ale rozšíriť ešte o jeden atribút, a to je poradové číslo pre daný typ fráze. Ako je zobrazené na obrázku 3.36, dátumy sú v tesnej blízkosti a ich vzájomné polohy voči ostatným sú takmer rovnaké. Ak by sa teda našiel podobný dokument s rovnakým rozložením, bolo by problematické namapovať fráze jedného dokumentu na fráze druhého. Jednalo by sa tu totiž o namapovanie jedného grafu a druhý (kde uzalmi sú fráze a hranami ich vzájomná poloha). Preto pre uľahčenie je treba priradiť indexy frázam.

Pridaďovanie indexov sa robí následovne:

1. *ImportantPhrase* sa rozdriedia do jednotlivých skupín podľa typu fráz
2. v každej skupine sa *ImportantPhrase* zoradia podľa polohy v dokumente (tj. tie, čo sú najbližšie pravému hornému rohu sú ako prvé)
3. každej *ImportantPhrase* sa priradí index podľa jej poradia v skupine. Tento index sa ukladá ako parameter *"occurence"*

Tento spôsob indexovanie zachováva indexy v prípade, keby jedena z *ImportantPhrase* by nebola v dokumente nájdená. (kebyže jedna z *ImportantPhrase* by nebola nájdená, rozhod to celé indexovanie a podobný dokument by tak nemusel byť nájdený)

V databázi nemusia byť uložené všetky vzájomné polohy. Napríklad ak fráza B je v oblasti UP fráze A, potom A bude v oblasti DOWN fráze B. Preto sa postupne prechádzajú fráze zhora nadol a ukladajú sa len dopredné fráze tak, ako je to zobrazené v kóde 3.13.

```
1 soretedPhreases = sort Important Phrases
2 for (i=0;i<soretedPhreases.size -1; i++){
```

```
3   for (j=i+1;i<soretedPhreases.size ;j++){
4      ulozVzajomnuPolohu(soretedPhreases[i],soretedPhreases[j])
5   }
6 }
```

Kód 3.13: Ukladanie vyájomných polôh

Prvá fráza funkcie *ulozVzajomnuPolohu()* sa označuje ako **ukladaná phrase**, a druhá sa označuje ako **referovaná fráza**.

V databázi sa vzájomná poloha ukladá v tabuľke *ssda_important_phrases* do týchto stĺpcov:

- phraseid - id ukladanej fráze
- documentid - id dokumentu
- phrasetype - typ *ukladanej fráze*
- occurence - index *occurence ukladanej fráze*
- area - oblasť, v ktorej sa nachádza *referovaná fráza* voči *ukladanej fráze*.
- referencedphrase - id *referovanej fráze*
- referencedtype - typ *referovanej fráze*
- referencedoccurence - index *occurence referovanej fráze*

Pre nájdenie rovakého dokumentu potom stačí v dokumente vyhľadať dokument, ktorý čo najviac zhodných *ImportantPhrases*. SQL dotaz pre nájdenie podobného dokumentu je zapísaný v kóde 3.14, kde číslo "1234567"je ID aktuálneho dokumentu:

```
1  SELECT ref.documentid, sum(1) as matchingrate
2  FROM ssda_important_phrases base ,ssda_important_phrases ref
3  WHERE
4    ref.phrasetype=base.phrasetype
5    and ref.referencedtype=base.referencedtype
6    and ref.occurence=base.occurence
7    and ref.referencedoccurence=base.referencedoccurence
8    and (base.area&ref.area > 0)
9    and base.documentid= 1234567
10 GROUP BY ref.docid
11 ORDER BY matchingrate
```

Kód 3.14: Vyhľadávanie podobných dokumentov

Ako je uvedené v kóde 3.14, porovnávajú sa fráze podľa typu, poradia fráze v dokumente (*occurence*), typu referenčnej fráze a jej poradia (jej *occurance*) a musí byť v rovnakej oblasti.

Dokument, ktorý je vyšetrovaný, čiže ten ku ktorému sa hľadá podobný dokument, sa označuje ako *base document*. V SQL dotaze je záznam z tabuľky *ssda_important_phrases* s aliasom *base*.

Výsledkom SQL dotazu je zoznam, ktorý obsahuje id dokumentu a počet zhodných prvkov s *base document*. Ako prvý dokument by mal byť **base document**, pretože najpodobnejší si je on sám. Jeho hodnota *matchingrate* udáva

maximálne skóre, ktoré je možné získať. Ak by však prvým dokumentom nebol *base document*, je jasné že nájdený dokument má maximálnu zhodu s *base document*. [29] V opačnom prípade sa vezeme druhý v poradí a jeho *matchingrate*. Tento dokument sa označuje ako **matching document**. Ak *matchingrate matching document*u je aspoň 80% z *matchingrate base document*u, potom sa tieto dokumenty automaticky označia za podobné. V opačnom prípade vyhľadávač dokumentov najprv napáruje nájdené *ImportantPhrases*, zobrazí ich užívateľovi s upozornením, na koľko sú dokumenty podobné a opýta sa či napárované *ImportantPhrases* sú v poriadku.

Je potrebné si uvedomiť, že do tabuľky sa ukladajú vzájomné polohy fráz "každý s každým", čiže ich kartézsky súčin. 80% zhoda (číselne 0,8) tohto kartézskeho súčinu teda ukazuje, že v $\sqrt{0,8} = 0.891$, teda v 89.1% frázach sú si dokumenty podobné.

V opačnom prípade, keď *matchingrate* nájdeného dokumentu menej ako 80%, vyhľadávač dokumentov najprv napáruje nájdené *ImportantPhrases*, zobrazí ich užívateľovi s upozornením, na koľko sú dokumenty podobné a opýta sa či napárované *ImportantPhrases* sú v poriadku.

3.5 Výhody riešenia

3.6 Nevýhody riešenia

OCR

Samozrejme najväčším problémom je OCR aplikácia, ktorá nedokáže previesť obrázok do textu tak dobre, ako človek. Problém prevodu obrázku do textu by sa dal zmenšiť kvalitnejším skenovaním dokumentov.

Sklon dokumentu

Ďalším faktorom je sklon dokumentu. Ak dokument formátu A4 je naklonený o viac ako 0.6 stupňa, očíslovanie *SpatialLine* a teda usporiadanie slov v dokumente, môže byť nesprávne, čo môže mať za následok, že v dokumente nebude správne rozoznaná tabuľka, ktorá zaberá celú šírku dokumentu.

Neúplné číselníky

Ak v číselníku chýba názov ulice, *Annotator* ju neidentifikuje a tým pádom celá adresa nemusí byť rozoznaná. Tá môže pre konkrétny typ dokumentu hrať kľúčovú rolu a dokument nemusí byť identifikovaný s tak veľkou presnosťou.

[29] Pretože PostgreSQL zaručuje len zoradenie podľa hodnoty *"matchingrate"*, môže sa stať, že prvý záznam nebude *base document*.

4. Užívateľská dokumentácia

4.1 Inštalácia

Aplikácia je určená pre operačný systém Windows verzie 7 s 64-bitovvým procesorom. Pre beh aplikácie je nutné mať nainštalovanú platformu Java verzie minimálne 8.

Z adresára *install* na priloženom CD je potrebné nainštalovať:

- platformu Java, ktorú je možné inštalovať zo súbora *jdk-8u25-windows-x64.exe*, alebo zo stránky *http://java.com/en/download*
- databázový systém PostgreSQL je možné nainštalovať zo súbora *postgresql-9.4.2-1-windows-x64.exe*, alebo zo stránky *http://www.postgresql.org/download/*
- Tesseract OCR, ktorú je možné nainštalovať zo súbora *tesseract-ocr-setup-3.02.02.exe*, alebo podľa návodu na stránke *https://github.com/tesseract-ocr/tesseract/wiki*

Pre jednoduhšiu manipuláciu s datazázou je dobré mať nainštalované klienta *PgAdminIII*. Ten je možné nainštalovať buď zo súboru *pgadmin3-1.20.0.zip*, alebo zo stránky *http://www.pgadmin.org/download/*.

V adresári *ssda* na priloženom CD je uložená aplkácia. Tú treba prekopírovať na harddisk, alebo médium, na ktoré je možné zapisovať. Aplikácia totiž svoje výstupné dáta zapisuje do adresára *out*.

Po inštalácií PostgreSQL je treba vytovriť databázu zo zálohy a nastaviť prístupové údaje pre aplikáciu. Záloha databáze je uložená v súbure *install/ssda-db.sql*.Prístupové údaje pre aplikíciu sa nastavujú v súbore *ssda/config/config.xml*.

4.2 Aplikácia

Aplikácia sa spúšťa príkazom "java -jar ssda.jar"Všetky súbory, ktoré sa majú zpracovať je potrebné nakopírovat do adresára *IN*, alebo uviesť súbory pre spracovanie ako argument príkazovej riadky.

Po spustení za zobrazí proces, ktorý aplikácia vykonáva. V prípade spracovania obrázkového súbora, alebo PDF súbora sa zobrazuje:

```
---- preprocessing files: 1 íýá
---- preprocessed files: 3.897s
---- loaded annotations config: 0.500s
OCR for 65023500-f97c-4abe-b1e4-ff200e5dec89.pdf_1.png: 0.0s
descewed: 1.116s
cutted: 8.481s
text recognized: 6.789s
total: 16.386s
```

Kód 4.1: Po spustení aplikácie

Výpis obsahuje informácie:

- koľko súborov bolo predspracovných

- ako dlho trvalo predspracovanie súbora
- inicializácia anotácií (nahranie konfigurácie pravidiel do pamäte zo súbora a z databáze)
- spustenie OCR aplikácie nad obrázkovým súborom *65023500-f97c-4abe-b1e4-ff200e5dec89.pdf_1.png*, ktorý vznikol z PDF súbora *65023500-f97c-4abe-b1e4-ff200e5dec89.pdf* v predspracovaní
- ako dlho trvalo detekcia a následné vyrovnanie obrázku
- ako dlho trvalo rozsekanie obrázku
- ako dlho trvalo spustenie OCR aplikácie nad rozsekanými obrázkami
- celkový čas strávené na prevode súboru do *SpatialDocument*u.

Po ňom nasleduje výpis anotovaných fráz:

```
1  BRUDER \& TEAM, Rastislav Bruder Iveta E Masarykova 493 90343 Kopč
       ↪ any (address,0.97)[(187, 233) - (1011, 362)]
2    Masarykova 493 (streetAddr,0.91)[(188, 282) - (440, 315)]
3      Masarykova (street,0.52)[(188, 282) - (374, 315)]
4      493 (houseNum,0.55)[(384, 283) - (440, 308)]
5    90343 Kopčany (cityzip,0.97)[(187, 330) - (444, 362)]
6      Kopčany (city,0.76)[(307, 330) - (444, 362)]
7      90343 (zip,0.94)[(187, 330) - (277, 355)]
8    BRUDER \& TEAM, Rastislav Bruder Iveta E (contactName,0.97)[(188,
         ↪ 233) - (1011, 270)]
9  Konštantný symbol: 0008 (constSymbol,0.85)[(1049, 329) - (1318, 352)
       ↪ ]
10   Konštantný symbol: (constSymbolLabel,0.50)[(1049, 329) - (1253,
         ↪ 352)]
11   0008 (constSymbolNum,0.91)[(1267, 331) - (1318, 348)]
12 Variabilný symbol: (varSymbolLabel,0.50)[(1047, 363) - (1238, 387)]
13 IČO (ICO,0.50)[(1049, 441) - (1089, 463)]
14 IČDPH: (ICDPH,0.50)[(187, 608) - (266, 631)]
15 DIČ: (DIC,0.50)[(187, 645) - (233, 668)]
16 Účet (accontNumberLabel,0.80)[(187, 705) - (238, 728)]
```

Kód 4.2: Výpís anotovaných fráz

Výpis obsahuje nájdené anotované fráze. Tie sú hierarchivky vypísané vo formáte: *text fráze (typ fráze, pravdepodobnosť)[(súradnice ľavého horného rohu) - (súradnice pravého dolného rohu)]* Z hiearchie fráz vo vzorovom kóde 4.2 je vidieť zloženie frázy typu adresa (*address*). Tá sa skladá z fráz typu *streetAddr*, *cityzip* a *contactName*.

Po tomto výpise aplikácia hľadá podobný dokument. V prípade ak podobný dokument nenájde, vyžiada si od užívateľa definovanie typu dokumentu, užívateľské označenie dokumentu (napríklad aký druh faktúry spracovaný dokument predstavuje) a následne sa začne dotazovať na jednotlivé políčka v dokumente, ako napríklad: *"ktoré z nájdených fráz je adresa"*. Príklad dotazovania je zobrazený v kóde 4.3.

```
1 Vyberte prosím typ aktuálneho dokumentu:
2 (0) Faktúra
3 (1) Poistenie
```

```
0
Zadajte prosím vlastné označenie dokumentu (napríklad meno poskytovateľa): BRUDER
Zvoľte prosím ktorá z uvedených fráz je "Dodávateľ":
(0) žiadna z uvedených
(1) BRUDER \& TEAM, Rastislav Bruder Iveta E Masarykova 493 90343 Kopčany
(2) Kollárova 318 90848 Kopčany
1
Zvoľte prosím ktorá z uvedených fráz je "Zákazník":
(0) žiadna z uvedených
(1) BRUDER \& TEAM, Rastislav Bruder Iveta E Masarykova 493 90343 Kopčany
(2) Kollárova 318 90848 Kopčany
2
Zvoľte prosím ktorá z uvedených fráz je "Položky":
(0) žiadna z uvedených
(1) 21.1.2011 DL21121016 Zľava v \%: 20\%DPH 2,35 0,00 ZD20\%DPH 11,73 0 \%: 10 \%: Základná sadzba 20 \%: SPOLU Základ 0,00 0,00 11,72 Výška DPH 0,00 0,00 2,35 Vrátane 091-1 0,00 0,00 14,07
1
Zvoľte prosím ktorá z uvedených fráz je "DIČ":
(0) žiadna z uvedených
(1) 1029229093
1
```

Kód 4.3: Dotazovanie na parametre neznámeho dokumentu

V prípade, ak aplikácia nájde podobný dokument, kde zhoda je väčšia ako 80%, automaticky vyplní typ dokumentu, užívateľské označenie dokumentu a priradí všetkym políčkam odpovedajúce fráze. Výslendé hodnoty zobrazí užívateľovi a dokument uloží do databáze. Ak by ale nájdený dokument mal menšiu zhodu ako 80%, aplikácia tiež vyplní typ dokumentu, užívateľské označenie dokumentu a priradí všetkym políčkam odpovedajúce fráze, ale upozorní na to užívateľa a dotáže sa ho, či sú vyplnené políčka správne. Ak užívateľ odpovie *"nie"*,aplikácia si vyžiada od užívateľa definovanie typu dokumentu, užívateľské označenie dokumentu, a tak ďalej, rovnako ako keby podobný dokument nebol nájdený. Časť príkladu je zobrazený v kóde 4.4.

```
...
Rozoznaný dokument je typu "faktura", s vlastným začením "bruder".
Nájdené položky:
DICNum: 1029229093
provider: Kollárova 318 90848 Kopčany
custommer: BRUDER & TEAM, Rastislav Bruder Iveta E Masarykova 493 90343 Kopčany
Dokument je zhodný len na 44%.
Sú nájdené položky v poriadku?[a/n]:
n
Vyberte prosím typ aktuálneho dokumentu:
(0) Faktúra
(1) Poistenie
0
Zadajte prosím vlastné označenie dokumentu (napríklad meno poskytovateľa): bruder
...
```

Kód 4.4: Oprava nesprávneho priradenia

5. Praktické experimenty

5.1 Vyhľadávanie slov v texte

Teoreticky by najlepšou dátovou štruktúrou mal byť regulárny výraz. To ale bolo pri testoch vyvrátené. Ako testovacie dáta boli vybrané 4 textové súbory, ktoré mali spolu 885 slov. Vyhľadávali sa slová zo zoznamov: mien, priezvisk a miest. Spolu mali 342280 slov. Pre každý zoznam sa vytvoril jeden regulárny výraz (text, ktorý odpovedal regulárnemu výrazu bolo poto možné označiť ako meno, priezvisko, alebo mesto). Celkový priemerný čas, vyhľadávania slov v súboroch bol 45.5 sekúnd. Pri vyhľadávaní konštánt v texte pomocou *HashMap*y, bolo nutné text rozdeliť na slová a tieto slová následne vyhľadať v *HashMap*e. Celkový priemerný čas vyhľadávania slov bol 0.21 sekúnd.

5.2 Pokusy a výsledky

Aplikácia bola trénovaná na ôsmich dokumentoch dokumentoch. Pri tom bolo treba:

1. vytvoriť nové pravidlo: (typ, text, pravdepodobnosť) : "city","h o l í č", 0.4

Závěr

Cieľom práce bolo navhnúť a implementovať algoritmus, ktorý dokáže analyzovať dokument, nájsť podobný a podľa neho priradiť určiť typ dokumentu, a priradí význam jednotlivým blokom textu.

Ako vstupné dokumenty boli použité oskenované dokumenty uložené v PDF súboroch, alebo obrázkoch. Preto bolo dôležité, aby text, rozoznaný z obrázkov bol v čo najlepšej kvalite. Vývoj modulu, ktorý rozrezával obrázky aby TesseractOCR vracal lepšie rozoznaný text v obrázku, bol časovo oveľa náročnejší než sa predpokladalo.

Pri testovaní a vývoji aplikácie sa prišlo na možné nedostatky, ktorých riešenia by mohli byť označené ako rozšírenie tejto práce:

- rozoznávanie je veľmi závislé na kvalitne rozoznanom texte, ktorý aby bol správne rozoznaný, musí byť uložený v pravidlách. Ako možné rozšírenie tejto práce by bolo navrhnúť spôsob, ako v dokumente vyhľadávať slová alebo regulárne výrazy s chybou.

Seznam použité literatury

[1] Fayez Tarsha-Kurdi, Tania Landes, Pierre Grussenmeyer. *Hough-transform and extended RANSAC algorithms for automatic detection of 3D building roof planes from Lidar data.*, Espoo, Finland, 2007, pp.407-412., halshs-00264843.

[2] Neznámi autor Anydoby, *Automatic image deskew in Java*. 27-10-2010, http://anydoby.com/jblog/en/java/1990

[3] Faisal Shafait, Daniel Keysers, and Thomas M. Breuel. *Performance Comparison of Six Algorithms for Page Segmentation*. 2006,D-67663, http://www.keysers.net/daniel/files/Shafait–Layout-Analysis-Comparison–DAS2006.pdf

[4] Canny J. *A Computational Approach To Edge Detection*. IEEE Trans. Pattern Analysis and Machine Intelligence, 1986, pp. 679–698

[5] Tom Gibara *Canny Edge Detector Implementation*. 2011,http://www.tomgibara.com/computer-vision/canny-edge-detector

[6] Shelley Powers *Practical RDF*. 2009, ISBN:978-0-596-00263-3

[7] Peter Buneman *Semistructured data*, Proc. ACM Symposium on Principles of Database Systems, pages 117-121, Tucson, AZ., 1997. Abstract of invited tutorial 2002, ISBN:1-4020-0489-3

[8] Josef KOLÁŘ *Teoretická informatika. 2. vyd.*, Praha : Česká informatická společnost, 2004. 205 s. ISBN 80-900853-8-5

Seznam obrázků

Seznam tabulek

Prílohy

	vlastnosť	LANA	Datamolino	ReadSoft Invoices
Predspracovanie	Používa descale obrázku	áno	?	áno
	Používa odstraňovanie šumu	nie	?	áno
OCR	Aké OCR používa	Tesseract	vlastné	Používa udajne 3, pre kontrolu správnosti
	Dokáže spracovať aj ručne písané znaky	nie	?	áno
	Podpora rôznych jazykov	áno	áno	áno
	Podpora rôznych fontov	áno	?	áno
	Je naučiť aplikáciu nový font	áno	?	nie
Spracovanie	Aký typ machine-learning algoritmu používa	prekrývanie blokov textu	?	anotácia textu, následne vyhľadá podľa odosielateľa šablónu dokumentu
	Dokáže software anotovať časti textu?	áno	?	áno
	dokáže software identifikovať obrázky na dokumente	nie	?	čiarkové kódy
	pracuje software s obrázkom	nie	?	áno
	extrakcia položiek	áno	áno	áno
	aký typy dokumentov rozoznáva	polo-štrukturované dokumenty	faktúry, proformy, účtenky	polo-štrukturované dokumenty
	extrakcia položiek	áno	áno	áno
	rozozná ten istý dokument v rôznych rozlíšeniach	áno	?	?
	dokáže spracovať viacstránkové dokumenty	nie	áno	áno
	dokáže identifikovať tabuľku v dokumente	nie	?	áno
	rýchlosť spracovania dokumentu	30 sek	3 min - 3 dni	6-12 sek

	je nutná spolupráca koncového užívateľa	áno	nie, všetko robí operátor	áno, ak pravdepodobnosť rozoznaných dát je menšia ako nastavená limita,
Výsledok	zobrazuje umiestnenie zroja dát na obrázku	áno	nie	áno
	umožňuje export dát	áno	áno	áno
	ako sa kontroluje kvalita výsledku	nijak	kontroluje operátor	logicky (napr. že sedia súčty), protiúčtové, alebo či sú správne údaje o dodávateľovi
	Kde sa vykonáva extrakcia dát	na vzdialenom serveri	na vzdialenom serveri	na vzdialenom serveri

Tabulka 5.1: Porovnanie vlastností existujúcich riešení